Modélisation hybride dynamique de flux de trafic

Rapport scientifique 2014

CISIT – Phase 5

Projet ISART

Hassane Abouaïssa, Yoann Kubera, Gildas Morvan

Univ Lille Nord de France, F-59000 Lille, France
UArtois, LGI2A, F-62400, Béthune, France

Résumé:

La simulation du trafic routier sur des réseaux de grande échelle est un problème compliqué car il suppose d'intégrer dans un même modèle différentes approches. Ainsi, les sections autoroutières sont généralement représentées à l'aide de modèles macroscopiques alors que pour les sections urbaines, des modèles microscopiques sont utilisés. De manière générale, les modèles microscopiques sont intéressants lorsque les interactions entre véhicules, ainsi que la topologie du réseau deviennent complexes.

Les modèles intégrant ces différents niveaux de représentation sont généralement qualifiés d'hybrides. Par ailleurs, ils sont généralement "statiques" : à chaque portion du réseau est associée une représentation unique qui ne changera pas au cours de la simulation. Afin de palier cette limitation, nous avons débuté en 2013 dans le cadre du projet ISART le développement d'un simulateur multi-agent multi-niveaux de flux de trafic routier nommé JAM-FREE permettant :

— de simuler des réseaux routier de grande taille efficacement en adaptant dynamiquement le niveau de détail,
— de tester de nouveaux algorithmes de régulation, d'observation et routage.

Ce simulateur repose sur un framework de modélisation et de simulation multi-agents multi-niveaux nommé SIMILAR (**SI**mulations with **M**ult**I**-**L**evel **A**gents and **R**eactions), implémenté en Java (Morvan et Kubera, 2014b) et distribué sous licence libre.

Dans ce rapport, nous présentons ces résultats scientifiques ainsi que les publications associées.

Table des matières

Chapitre 1

Introduction

Cet article rapporte les travaux conduits par Hassane Abouaïssa [1], Yoann Kubera [2] and Gildas Morvan [3] durant la phase 5 de CISIT [4] dans le projet ISART.

Contexte

Un flux d'agent mobiles peut être observé à différentes échelles. Ainsi dans la modélisation du flux de trafic routier, trois niveaux sont généralement considérés : *micro*, *méso* et *macro*, représentant respectivement les interactions entre véhicules, groupes de véhicules partageant certaines propriétés (comme une position voisine ou une même destination) et enfin flux de véhicules. Chaque approche est utile dans un contexte particulier : les modèles micro et méso permettent de simuler des réseaux à topologie complexe comme les zones urbaines, alors que les modèles macro permettent de développer des stratégies de contrôle pour prévenir les congestions sur les autoroutes.

Cependant, pour simuler des réseaux routiers de grande taille, il peut être intéressant d'intégrer ces différentes représentations dans un même modèle comme le montre la fig. 1.1.

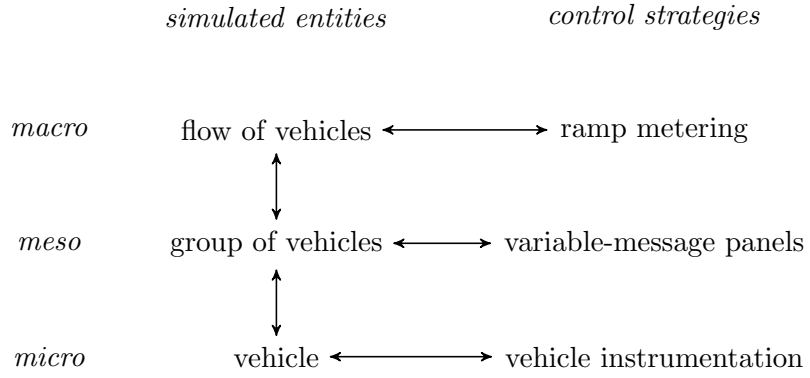


FIGURE 1.1 – Simulation et contrôle hybride de flux de trafic

Dans de telles approches, le réseau routier est découpé en plusieurs parties. Dans chaque partie, le flux de trafic est simulé à l'aide de son modèle propre modèle (micro, méso ou macro). La

1. http://www.lgi2a.univ-artois.fr/spip/spip.php?article5&idpersonne=5
2. http://www.yoannkubera.net/
3. http://www.lgi2a.univ-artois.fr/~morvan/
4. http://www.cisit.org/

cohérence générale de la simulation est assurée par l'interconnexion entre les diverses parties et l'échange de données telles que la densité, la vitesse ou le flux moyen des véhicules. *Ces modèles hétérogènes sont appelés modèles hybrides.* Certains modèles hybrides, associant représentations micro and macro sont présentés dans le tableau 1.1.

modèle	micro	macro
Magne *et al.* (2000)	SITRA-B+	SIMRES
Poschinger *et al.* (2002)	IDM	Payne
Bourrel et Lesort (2003)	optimal velocity	LWR
Mammar et Haj-Salem (2006)	optimal velocity	ARZ
Espié *et al.* (2006)	ARCHISM	SSMT
El hmam (2006)	generic ABM	LWR, ARZ, Payne

TABLE 1.1 – Principaux modèles hybrides de flux de traffic, adapté de El hmam (2006, p. 42)

Motivations

Les modèles hybrides présentés dans le tableau 1.1 semblent fournir le compromis nécessaire à la simulation de réseaux routiers de grande taille. Toutefois, ils partagent la même limitation : les connexions entre niveaux sont fixées *a priori* et ne peuvent être changées durant la simulation. Ils se révèlent ainsi incapables d'observer avec suffisamment de détails certains phénomènes émergents comme la formation et le suivi de congestions (voir figures 1.2 et 1.3).

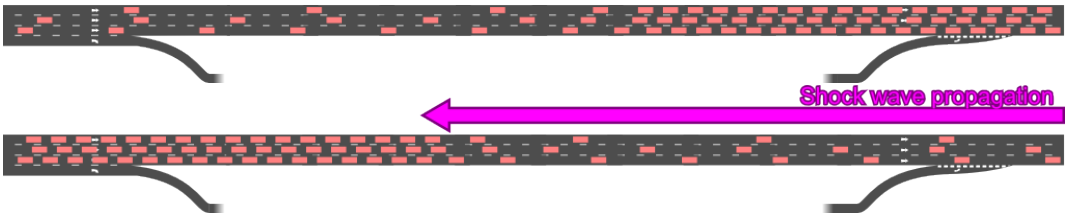


FIGURE 1.2 – Phénomène d'onde de choc, provoquant le déplacement vers l'amont d'une congestion

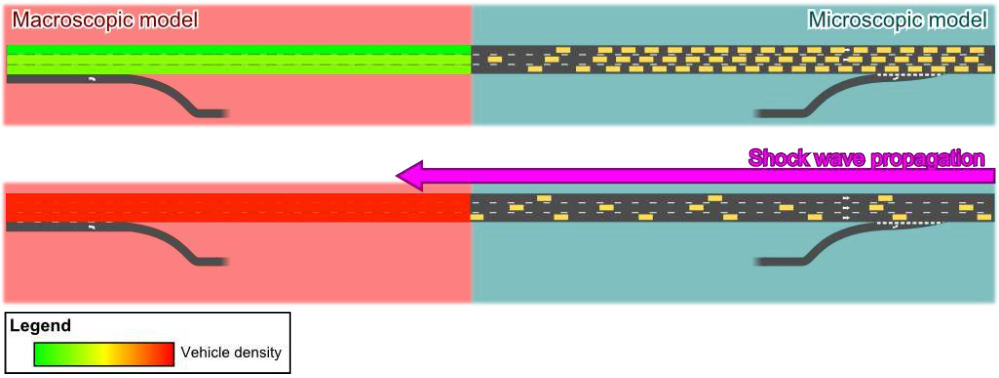


FIGURE 1.3 – Modèle hybride *"statique"* micro/macro dans le cas d'une onde de choc (voir figure 1.2)

Pour observer de tels phénomènes, le découpage ne doit pas être *statique* mais *dynamique* (Morvan et Kubera, 2014b; Sewall *et al.*, 2011) autant du point de vue spatial et de l'échelle. En effet, le suivi d'ondes de choc nécessite le redimensionnement à la volée d'une zone micro (fig 1.4) et donc un modèle hybride dynamique de l'espace. De même, l'étude de l'apparition d'une congestion dans une zone simulée avec un modèle macro nécessite un changement d'échelle à un niveau micro et donc un modèle hybride dynamique d'échelles.

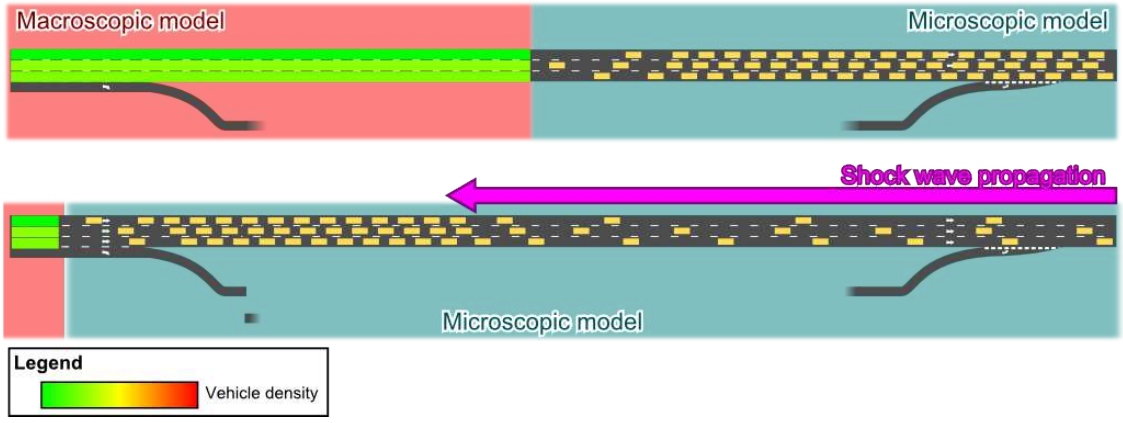


FIGURE 1.4 – Modèle hybride dynamique micro/macro d'une onde de choc (voir figure 1.2)

La modélisation multi-agents multi-niveaux est une approche intéressante pour simuler de tels systèmes (Gaud *et al.*, 2008; Gil-Quijano *et al.*, 2010; Gil-Quijano *et al.*, 2012; Picault et Mathieu, 2011; Vo *et al.*, 2012b,a; Vo, 2012). En effet, ces travaux proposent une large gamme de techniques pour adapter dynamiquement le niveau de détail des simulations, intégrer des modèles hétérogènes ou détecter et réifier des phénomènes émergents (Morvan, 2013).

Ainsi, en 2013, nous avons commencé le développement d'un simulateur multi-agents multi-niveaux appelé JAM-FREE dans le cadre du projet ISART. Il permet de simuler les grands réseaux routiers efficacement en utilisant un niveau de détail dynamique. Ce simulateur repose sur un framework de modélisation baptisé SIMILAR — appelé anciennement IRM4MLS (Morvan *et al.*, 2011; Morvan et Jolly, 2012; Soyez *et al.*, 2013) — pour **SI**mulations with **M**ult**I**-**L**evel **A**gents and **R**eactions.

Les chapitres suivants présentent les deux délivrables que nous avons développés :

— SIMILAR

— JAM-FREE

Chapitre 2

SIMILAR

Résumé:

La simulation de systèmes complexes nécessite souvent l'intégration de connaissances provenant de différentes sources pour obtenir des résultats pertinents. Ces sources peuvent être des personnes dont les champs d'application diffèrent ou plus simplement de points de vue différents sur le même phénomène.

Pourtant, les méta-modèles classiques pour la simulation multi-agents ne sont pas conçus pour intégrer de tels systèmes : leur représentation des agents, de l'environnement et de la dynamique temporelle du système repose sur un seul point de vue.

Pour répondre à ce problème, nous avons développé une approche générique appelée SIMILAR pour concevoir des simulations (fig. 2.1). Cette approche repose une représentation multi-agents multi-niveaux fondée sur le principe influence/réaction afin d'intégrer ces différents points de vue (Ferber et Müller, 1996; Michel, 2007a,b; Morvan *et al.*, 2011; Morvan et Jolly, 2012; Soyez *et al.*, 2013). Cette approche comprend un modèle formel générique, une méthodologie et une API de simulation qui préserve la structure et la sémantique tu modèle formel (Morvan et Kubera, 2014b). Ces éléments sont disponibles en ligne sur le site `http://www.lgi2a.univ-artois.fr/~morvan/similar.html`.

FIGURE 2.1 – Logo de SIMILAR

2.1 Travaux connexes

De nombreux méta-modèles dédiés à la simulation multi-agents multi-niveaux ont été proposés dans la littérature. Une étude complète peut être trouvée dans Morvan (2013). On notera que ces travaux imposent généralement des contraintes fortes sur les modèles (par exemple sur la structure des graphes d'interaction ou d'influence). À l'opposé, SIMILAR permet de simuler tout type de modèle multi-niveaux car il repose sur un modèle formel d'intéraction (Ferber et Müller, 1996).

2.2 Fondements de SIMILAR

Le modèle théorique sur lequel repose SIMILAR est une extension du modèle IRM4MLS (Morvan *et al.*, 2011; Morvan et Jolly, 2012) introduisant plusieurs notions clarifiant les problèmes d'interaction entre niveaux fonctionnant avec des temporalités différentes. Ces apports théoriques incluent l'ajout de notions et d'algorithmes relatifs à :

- l'état de la simulation, pouvant à tout moment être dans un état :
  - "stable" (*"consistent"* en anglais), si tous les niveaux ont fini d'évaluer le comportement des agents et de l'environnement qu'ils contiennent ;
  - "transitoire" (*"transitory"* en anglais), si tous les niveaux sont actuellement en train d'évaluer le comportement des agents et de l'environnement qu'ils contiennent ;
  - "partiellement stable" (*"half-consistent"* en anglais), dans les autres cas
- la *"désambiguation"* d'état, permettant d'estimer l'état de la simulation si jamais certains niveaux sont dans un état transitoire ;
- la structure de l'état des agents et de l'environnement ;
- la notion d'*"influence système"*, modélisant une influence que l'on peut trouver dans tout type de simulation.

2.3 Architecture de SIMILAR

L'approche SIMILAR et son implémentation ont été conçues afin de fournir le meilleur compromis possible entre performances et réutilisabilité de ses composants. Pour cela, une simulation peut être implémentée à l'aide de deux noyaux différents :

- Un *"micro noyau"* fournissant une implémentation minimale de SIMILAR. Cette implémentation offre de grandes possibilités d'optimisations ;
- Un *"noyau étendu"* fournissant une implémentation générique et modulaire des composants d'une simulation, permettant ainsi de mieux compartimenter et réutiliser les composants d'une simulation.

2.3.1 Le micro noyau de SIMILAR

SIMILAR est basé sur le concept de micro noyau, qui définit les classes principales permettant d'effectuer des simulations. Il fournit une correspondance directe entre les concepts théoriques et l'implémentation concrète d'une simulation. Il ne fournit que l'implémentation minimale de ces concepts, laissant ainsi au modélisateur une grande liberté. Le micro noyau de SIMILAR est constitué de trois composants (fig. 2.2) :

- L'API du micro noyau, contenant les classes Java de ce noyau.
- Les libraries standard de ce noyau minimal, contenant des implémentations génériques des algorithmes de simulation et des outils d'export des résultats.
- Les exemples illustrant l'utilisation du micro noyau et des common libraries pour concevoir des simulations.

2.3.2 Le noyau étendu de SIMILAR

Le noyau étendu fournit toute une classe d'outils facilitant la conception, la modification et l'extension de simulations. Le noyau étendu de SIMILAR est constitué de trois composants :

- L'API du noyau étendu, contenant les classes Java de ce noyau.

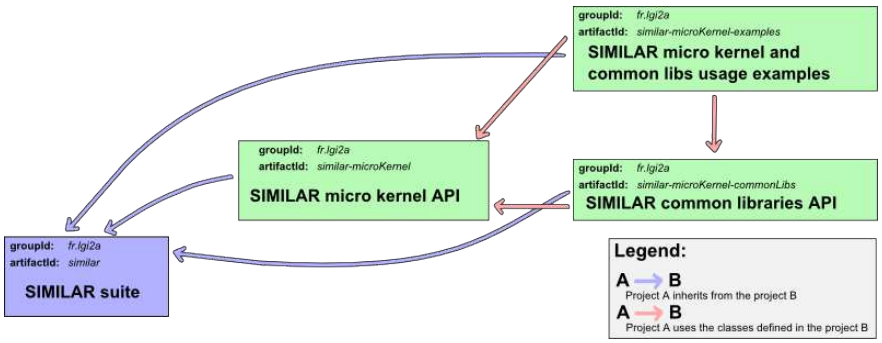


FIGURE 2.2 – Structure du micro noyau de SIMILAR

— Les libraries étendues de ce noyau, contenant des implémentations génériques des outils de conceptions définis par le noyau.
— Les exemples illustrant l'utilisation du noyau étendu et des libraries étendues pour concevoir des simulations.

2.3.3 Combiner les noyaux

Le micro noyau et le noyau étendu de SIMILAR sont inter-compatibles. Il est en effet tout à fait possible de faire cohabiter dans une même simulation des éléments provenant du micro noyau avec des éléments provenant du noyau étendu (que cela soit des agents, des environnements ou des niveaux). Cette flexibilité permet aux utilisateur de trouver le compromis entre performance et réutilisabilité correspondant le mieux à leurs attentes.

2.4 Anatomie d'une simulation

SIMILAR est conçu pour avoir une architecture hautement personnalisable, tout en rendant la structure des simulations aussi explicite que possible. Chaque concept de SIMILAR est intégré dans une classe concrète. A haut niveau, cela se traduit par la séparation entre les éléments constitutifs d'une simulation (fig. 2.3) :

— Le modèle de simulation (*i.e.* "simulation model"), contenant la partie déclarative de la simulation (la description du phénomène simulé) ;
— Le moteur de simulation (*i.e.* "simulation engine"), contenant la partie procédurale de la simulation (algorithmes génériques permettant d'exécuter une simulation) ;
— Les outils d'observation (*i.e.* "observation probes"), capables de récupérer des informations sur la simulation et de les exporter dans divers formats.

2.4.1 Le moteur de simulation

Dans SIMILAR, la notion de modèle de simulation est séparée de la notion de moteur de simulation pour distinguer les connaissances déclarative et procédurale de la simulation :

— Le modèle de simulation contient la définition des niveaux, les agents, l'action naturelle de l'environnement et la réaction de chaque niveau. Cette connaissance est spécifique au domaine.
— Un moteur de simulation assemble les informations liées à l'exécution d'une simulation. Il définit l'évolution du temps, quand demander aux agents de percevoir, de mémoriser ou

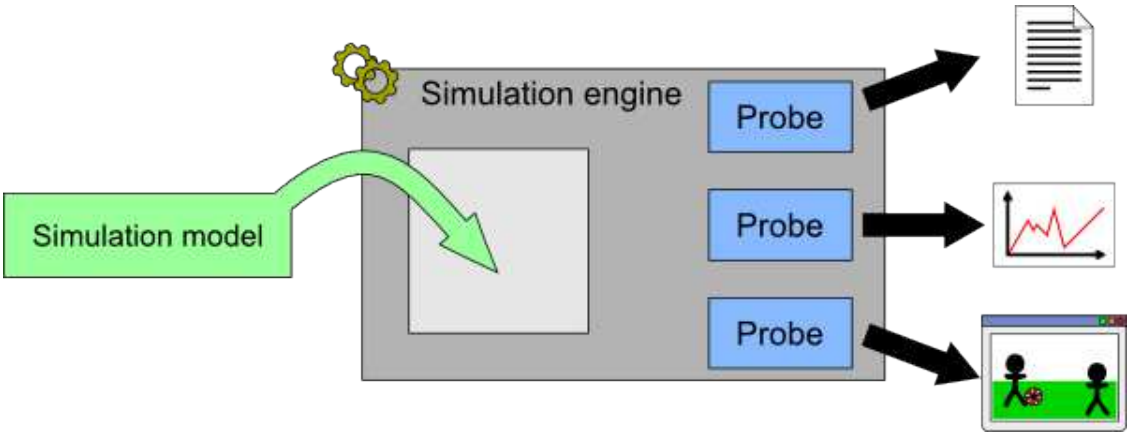


FIGURE 2.3 – Architecture de SIMILAR

décider, et détermine le moment où l'environnement produit son action naturelle ou encore le moment où la réaction est effectuée. Il est également responsable de l'observation et de l'exportation des données de simulation, ainsi que de la réaction aux influences système (création d'un nouvel agent, changement de niveau, etc.).

La séparation entre le modèle et le moteur facilite l'optimisation des simulations : le mode d'exécution le plus approprié peut être choisi en fonction de la simulation. En effet, un moteur de simulation peut être implémenté à l'aide de différents mécanismes internes pour gérer l'exécution d'une simulation.

Le moteur de simulation joue des rôles différents dans la simulation :

— Faire évoluer le temps de façon appropriée et gérer :
  — les phases de perception, mémorisation, décision des agents ;
  — l'influence naturelle de l'environnement ;
  — la phase de réaction des niveaux ;
— Assurer que les contraintes liées au modèle temporel restent valides durant toute l'exécution de la simulation. Morvan *et al.* (2011) décrit l'ensemble de ces contraintes ;
— Assurer que les probes d'observation sont mis à jour de façon appropriée ;
— Fournir une réaction aux influences système pour n'importe quel simulation.

Considérer le moteur de simulation comme une abstraction de haut niveau permet facilement de simuler un même modèle sur différentes architectures de calcul comme les CPUs multi-coeurs ou les GPGPU (general purpose graphical processing unit) comme le montre la fig. 2.4.

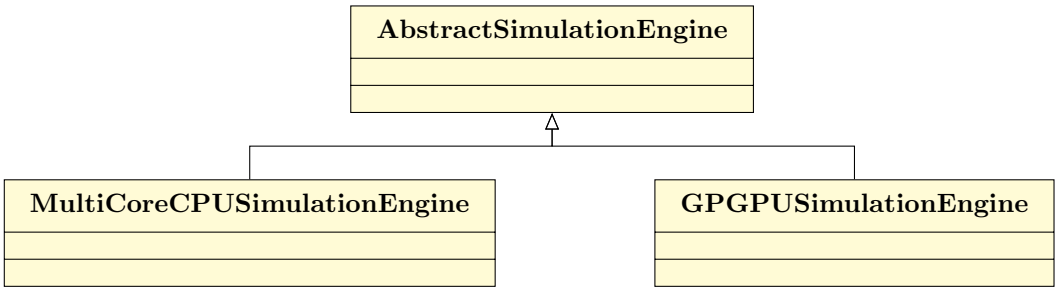


FIGURE 2.4 – Moteurs de simulation de SIMILAR

2.4.2 Probes

L'observation des données de la simulation est gérée au moyen de probes. Les probes sont des objets écoutant l'évolution de l'état de la simulation. Comme l'état de la simulation n'est pas toujours cohérent (en particulier lors du calcul de la réaction), les probes ne décident pas par eux-mêmes quand observer la simulation. Ainsi, ils sont liés à l'état du moteur de simulation. Le moteur de simulation est responsable de dire aux probes lorsque l'état de la simulation est cohérent, et donc quand les probes peuvent observer la simulation et exporter des résultats.

2.5 Livrables et publication des résultats

L'implémentation de SIMILAR est actuellement disponible en ligne sur le site de Gildas Morvan [1]. Elle est diffusée sous la licence de logiciel libre CeCILL-B [2] (voir figure 2.5).

```
This software is a computer program whose purpose is to support the
implementation of multi-agent-based simulations using the formerly named
IRM4MLS meta-model. This software defines an API to implement such
simulations, and also provides usage examples.

This software is governed by the CeCILL-B license under French law and
abiding by the rules of distribution of free software.  You can  use,
modify and/ or redistribute the software under the terms of the CeCILL-B
license as circulated by CEA, CNRS and INRIA at the following URL
"http://www.cecill.info".

As a counterpart to the access to the source code and  rights to copy,
modify and redistribute granted by the license, users are provided only
with a limited warranty  and the software's author,  the holder of the
economic rights,  and the successive licensors  have only  limited
liability.

In this respect, the user's attention is drawn to the risks associated
with loading,  using,  modifying and/or developing or reproducing the
software by the user in light of its specific status of free software,
that may mean  that it is complicated to manipulate,  and  that  also
therefore means  that it is reserved for developers  and  experienced
professionals having in-depth computer knowledge. Users are therefore
encouraged to load and test the software's suitability as regards their
requirements in conditions enabling the security of their systems and/or
data to be ensured and,  more generally, to use and operate it in the
same conditions as regards security.

The fact that you are presently reading this means that you have had
knowledge of the CeCILL-B license and that you accept its terms.
```

FIGURE 2.5 – Extrait de la Licence Cecill-B utilisée dans la distribution de SIMILAR.

1. http://www.lgi2a.univ-artois.fr/~morvan/similar.html
2. http://www.cecill.info/licences/Licence_CeCILL-B_V1-fr.html

Le site propose également de récupérer le code source de SIMILAR, d'y lire la documentation ou de consulter l'API des noyaux de SIMILAR (voir 2.6).

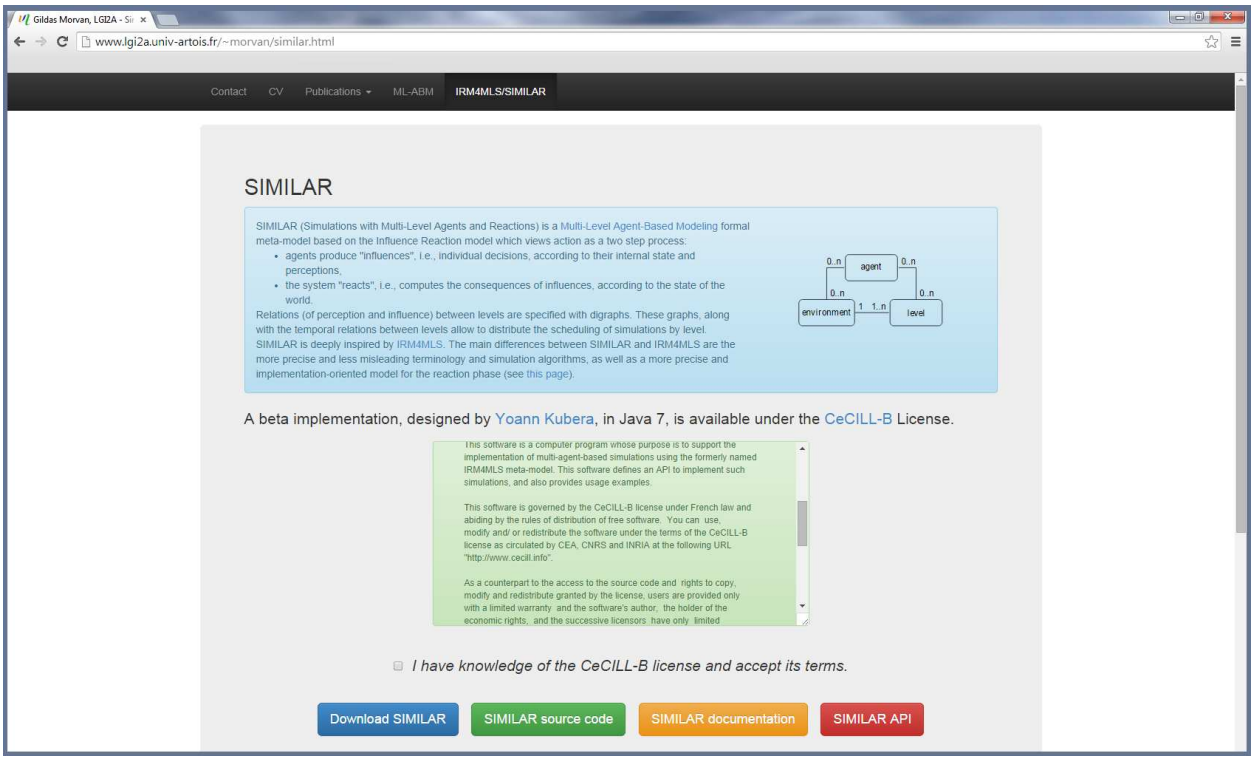


FIGURE 2.6 – Page web de diffusion de SIMILAR

Chapitre 3

JAM-FREE

Résumé:

Dans cette section, nous présentons le modèle hybride dynamique de flux de traffic appelé JAM-FREE – pour "Java, Agent and Multi-level based Framework for Road-traffic Examination and Enhancement" (voir figure 3.1) – basé sur SIMILAR.

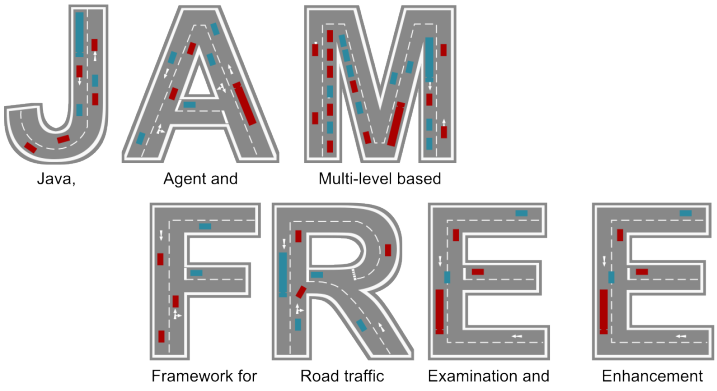


FIGURE 3.1 – Logo de JAM-FREE

3.1 Motivations et travaux connexes

3.1.1 Motivations

Nos motivations pour développer JAM-FREE sont :

1. simuler le trafic routier sur de très grandes zones comprenant à la fois les villes et les autoroutes,
2. être en mesure de tester différentes stratégies de régulation ou de redirection du trafic,
3. évaluer leur influence précise sur la fluidité de la circulation, et
4. comprendre pourquoi des perturbations de la circulation se produisent.

Le premier objectif est facilement atteint en utilisant des modèles macroscopiques au détriment du quatrième but. A l'inverse, le quatrième objectif est facilement atteint en utilisant des modèles

microscopiques au détriment du premier objectif. Pour faire face à ce paradoxe, nous avons conçu un modèle hybride dynamique permettant de bénéficier des deux approches.

Les avantages de cette approche hybride comprennent la capacité :

- d'obtenir des informations quantitatives et qualitatives sur la circulation routière, en utilisant des représentations de simulation respectivement macroscopiques et microscopiques dans la même simulation.
- de basculer entre ces représentations localement :
  - Pour les besoins de la simulation. Par exemple comprendre la source d'un embouteillage.
  - Pour gérer les contraintes de calcul. Par exemple la gestion de la charge du processeur.
- expérimenter des stratégies de routage macroscopiques et microscopiques, i.e., des stratégies d'équilibrage de la charge sur différentes routes.

3.1.2 Travaux connexes

A notre connaissance, le seul travail décrivant un modèle de ce type est (Sewall *et al.*, 2011) [1]. Cependant, les auteurs mettent l'accent sur la visualisation de données plutôt que la simulation précise des flux de trafic.

3.2 Cas d'utilisation

Dans le cadre de ce projet, nous avons défini une dizaine de cas d'utilisation pouvant être traités uniquement par des modèles hybrides dynamiques, et donc par JAM-FREE. Dans cette section, nous en présentons très brièvement deux. Une description plus étoffée de ces cas d'utilisation est fournie dans l'annexe B page 31 de ce document.

Le modèle hybride est capable de changer dynamiquement la représentation de la circulation sur une partie du réseau routier. Cette fonction peut être utilisée pour passer :

- d'une représentation microscopique à une représentation macroscopique lorsque le CPU est surchargé, dans le but de réduire sa charge ;
- d'une représentation macroscopique à une représentation microscopique pour connaitre la raison pour laquelle un embouteillage est apparu.

3.3 Modèle hybride dynamique

Dans JAM-FREE, le réseau routier est divisé dynamiquement en sous-ensembles que nous nommons "clusters" (voir fig. 3.2). Le trafic routier est simulé sur chaque cluster à l'aide de modèles hétérogènes dépendant du contexte. Ces derniers peuvent être par exemple soit des modèles microscopiques, soit des modèles macroscopiques (voir fig. 3.3).

Le modèle hybride dynamique de JAM-FREE repose sur un modèle du réseau routier et du flux de traffic à différents niveaux de représentation :

- Un niveau *"Infrastructure"* détaillant avec précision le réseau routier tel qu'il existe. Cette représentation inclut les boucles de détection de véhicules, ainsi que les données réelles ayant été récupérées à ces endroits ;
- Un niveau *"Contrôle"* gérant le découpage du réseau routier en *clusters*. Chaque cluster simule le flux de trafic en utilisant son propre modèle.
- Divers niveaux dédiés à la simulation du trafic routier selon un modèle particulier (niveau "Microscopique", niveau "Macroscopique", niveau "Mesoscopique", niveau "UTMC", *etc.*)

1. Les auteurs maintiennent une page web liée à leur article : `http://gamma.cs.unc.edu/HYBRID_TRAFFIC/`

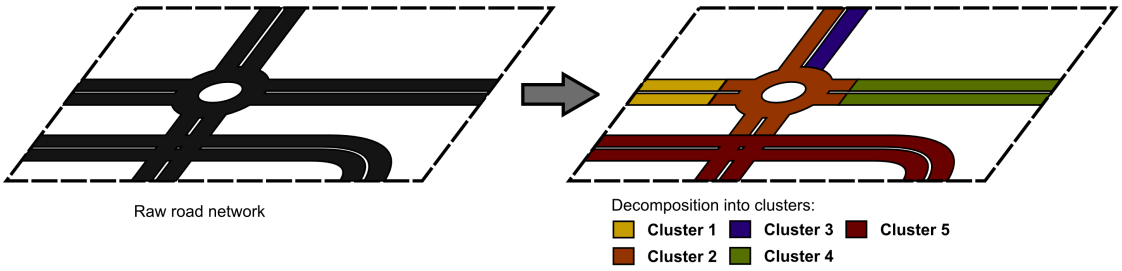


FIGURE 3.2 – JAM-FREE road network structure

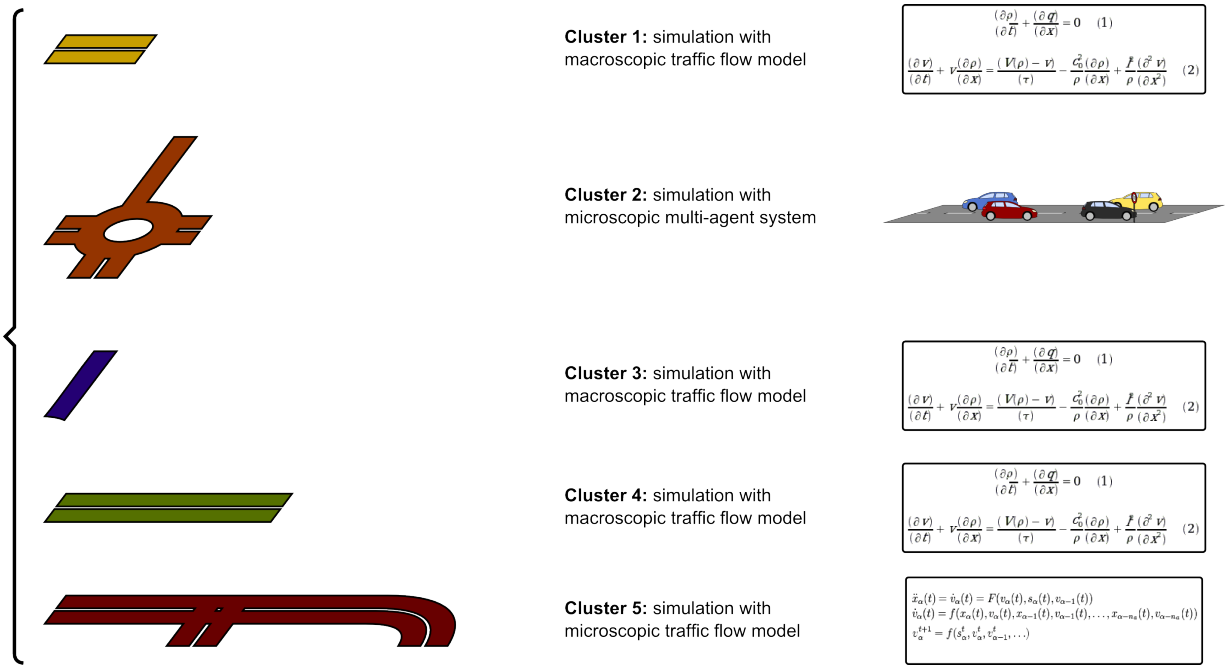


FIGURE 3.3 – JAM-FREE traffic representations.

Chaque niveau dispose de sa propre représentation du réseau routier et du flux de trafic, afin de répondre à son rôle dans la simulation. Dans cette section, nous fournissons un aperçu de ces différents modèles. Nous illustrons pour cela nos divers concepts dans le cas d'une simulation effectuée sur une autoroute avec une voie d'insertion (voir figure 3.4).

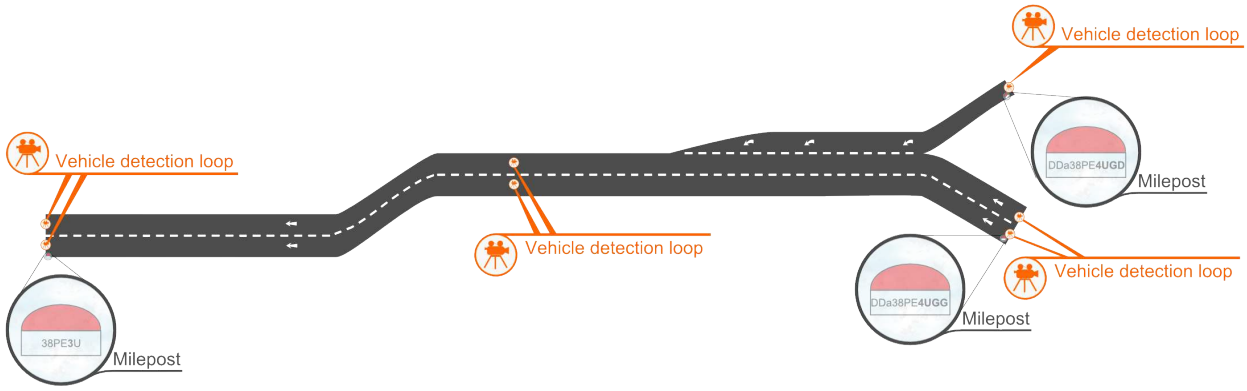


FIGURE 3.4 – Le réseau routier utilisé pour illustrer les concepts de JAM-FREE.

3.4 Le niveau "Infrastructure"

Pour des raisons d'efficacité, nous choisissons de ne pas modéliser le réseau routier à un niveau physique. En effet, un tel modèle nécessite d'intégrer aux véhicules des perceptions complexes. Ces calculs peuvent être évités, puisque dans notre cas d'utilisation les conducteurs sont généralement rationnels et conduisent en respectant les voies signalées au sol. Ainsi, le réseau routier n'est modélisé qu'à un niveau sémantique.

3.4.1 Modèle général

Au niveau "Infrastructure", la dénomination, l'identification et la représentation des composants de réseau routier suivent les spécifications de la Sétra [2] [3] (Sétra, 2010) :

— Le réseau routier est représenté sous la forme d'un graphe, dont les noeuds sont des Point LOcalisants [4] (PLO) situés à chaque changement de géométrie des routes (carrefour, rond de point, voie d'insertion, *etc.*). Les arcs modélisent les routes (voir figure 3.5) ;

— Une position dans le réseau est un triplet composé d'un PLO, d'une abscisse curviligne relative à ce PLO (dans le sens croissant des numérotations de PLO) ainsi que d'un identifiant de voie, représenté dans notre cas par un nombre entier (voir figures 3.6 et 3.9).

2. `http://www.setra.developpement-durable.gouv.fr`
3. `http://dtrf.setra.fr/pdf/pj/Dtrf/0005/Dtrf-0005792/DT5792.pdf`
4. Généralisation de la notion de Point Kilométrique PK et Point de repère PR

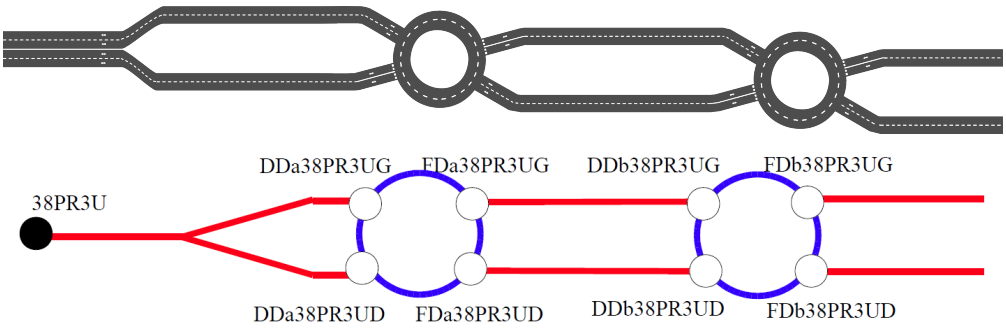


FIGURE 3.5 – Représentation d'un réseau routier en graphe, selon les spécifications de la Sétra.

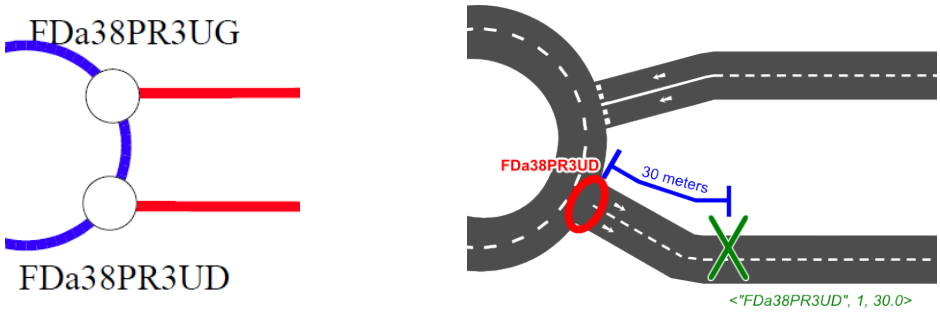


FIGURE 3.6 – Représentation d'une position dans le réseau routier, selon les spécifications de la Sétra.

Dans JAM-FREE, le réseau routier est représenté comme un ensemble de composants modélisant explicitement chaque élément du réseau routier (voir figure 3.7), que cela soit :

- les voies (composants bleu foncé "Lane component" de la figure 3.7)
- les routes (composants oranges "Road component" de la figure 3.7)
- les boucles de détection de véhicules (composants violets "Sensor component" de la figure 3.7)
- les débuts et fins de voies (composants rouge foncés "Lane end component" de la figure 3.7)
- le début et la fin du réseau simulé (composants gris "Network start component" et "Network end component" de la figure 3.7)
- le réseau routier simulé (composant bleu clair "Network component" de la figure 3.7)
- ce qui est extérieur au réseau simulé (composant vert foncé "Outside component" de la figure 3.7)

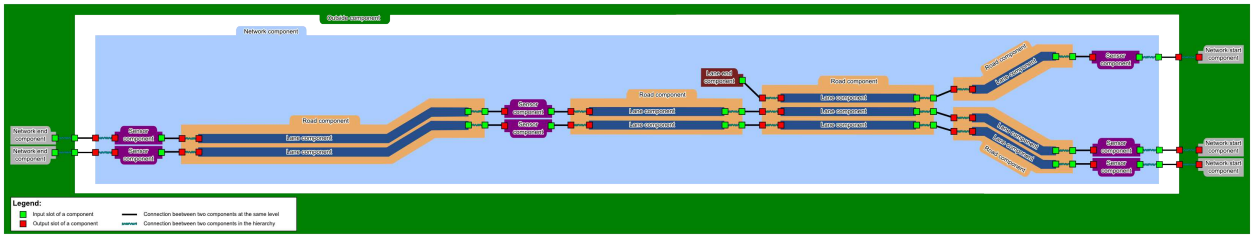


FIGURE 3.7 – Illustration d'un modèle de réseau routier correspondant à celui de la figure 3.4, du point de vue du niveau "Infrastructure". Une version agrandie de cette image peut être consultée à la page 30

Ces composants sont à la fois interconnectés de manière horizontale (*e.g.* une route avec une route voisine) et verticale (*e.g.* de contenu à conteneur, par exemple entre une voie et une route).

3.4.2 Signalisation routière

Dans notre modèle, nous intégrons un sous-ensemble des panneaux de signalisation français, pertinents pour notre approche (voir figure 3.8). A ce jour, ces panneaux incluent :

— les panneaux de limitation de vitesse,
— les panneaux de fin de limitation de vitesse,
— les panneaux d'interdiction d'emprunter une voie
— les panneaux d'autorisation d'emprunter une voie

Il est également possible d'ajouter des panonceaux à ces contraintes de conduites, afin de restreindre leur effet à certaines catégories de véhicules. Ces panonceaux incluent actuellement :

— la restriction à une catégorie de véhicule ;
— la restriction à des véhicules ayant un poids particulier ;
— la restriction à des véhicules ayant une hauteur particulière ;
— la restriction à des véhicules ayant une largeur particulière.

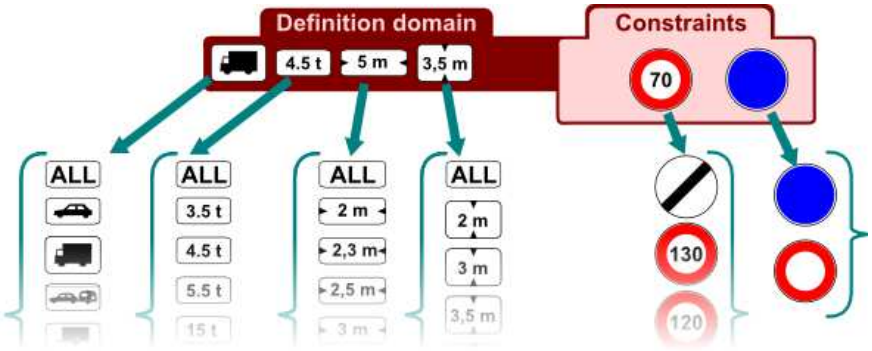


FIGURE 3.8 – Illustration du modèle de signalisation routière dans le niveau "Infrastructure".

3.4.3 Voies

Dans JAM-FREE, les composant modélisant les voies contiennent un grand nombre d'informations concernant le réseau routier. Elles sont caractérisées par :

— les coordonnées dans le réseau de leur point d'entrée/de sortie, dans le sens normal de circulation (voir figure 3.9) ;
— leur forme dans le monde réel, principalement pour des raisons d'affichage. Cette forme est représentée sous la forme d'une largeur et d'une suite de coordonnées, en mètres, relativement à une origine arbitraire (voir figure 3.10) ;
— les panneaux qu'elles contiennent. La position des panneaux est modélisée sous la forme d'une abscisse curviligne dans la voie (voir figure 3.11).

3.4.4 Boucles de détection de véhicules

Les boucles de détection jouent un rôle important dans le modèle d'une simulation à plusieurs titres. Outre leur rôle fondamental dans la construction des clusters (voir section 3.5 page 20), elles

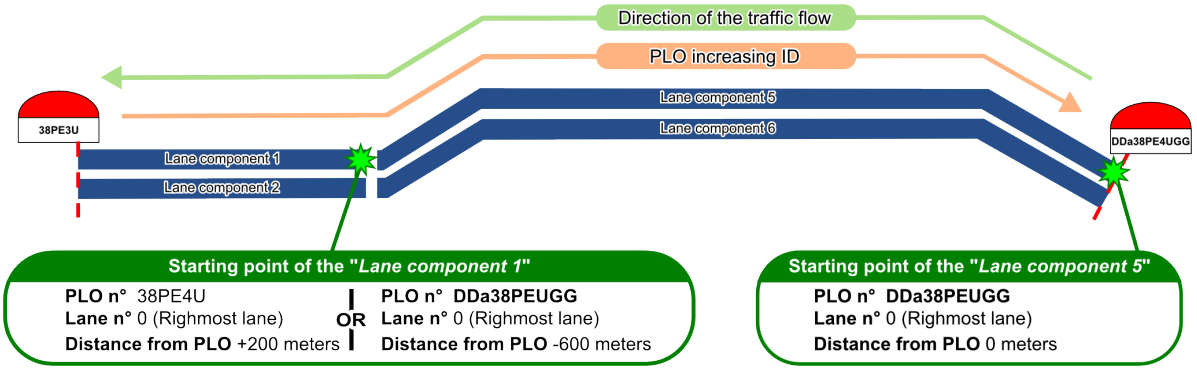


FIGURE 3.9 – Illustration de la notion de point d'entrée/de sortie d'une voie dans le niveau "Infrastructure".

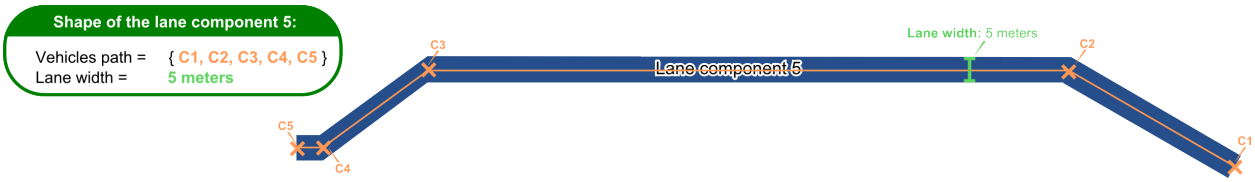


FIGURE 3.10 – Illustration de la notion de forme d'une voie dans le niveau "Infrastructure".

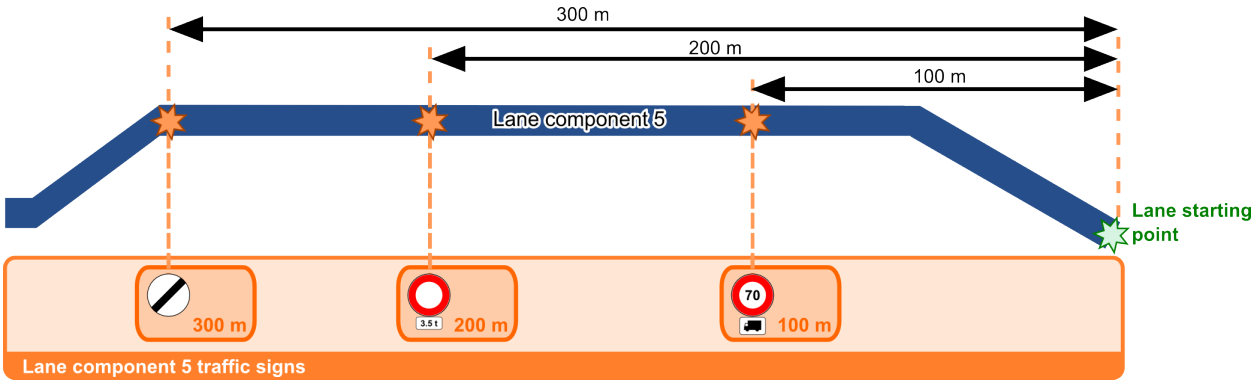


FIGURE 3.11 – Illustration du placement des panneaux dans une voie dans le niveau "Infrastructure".

font le lien entre le monde réel et le monde simulé, en fournissant les données réelles qui alimenteront la simulation.

Dans JAM-FREE, ces données contiennent deux valeurs : le flux moyen de véhicules, en vehicules par minutes ou véhicules par secondes, et la vitesse moyenne aggrégée des véhicules, en mètres par secondes ou kiolmètres par heure. L'unité utilisée est déterminée lors de la déclaration des boucles de détection dans le modèle de la simulation (voir section 3.7.5).

Deux formats de données réelles sont actuellement supportés :

— des données statiques, qui n'évolueront pas au cours du temps ;

— des données dynamiques définies dans un fichier ODS. Ce fichier définit le temps d'échantillonnage des données (qui peut être irrégulier) ainsi que la valeur du flux de véhicules et de leur vitesse moyenne pour chaque échantillon.

Les données sont définies individuellement pour chaque boucle de détection de véhicules, sous la forme de fichiers séparés, simplifiant ainsi la modification locale de données. Un exemple d'un tel fichier est fourni dans l'annexe C page 36.

3.5 Le niveau "contrôle"

Le niveau "Contrôle" de la simulation constitue le cœur de l'approche hybride dynamique. Il permet de gérer dynamiquement le découpage du réseau routier en sous-parties que nous appelons *cluster*. Chaque cluster fonctionne de manière autonome, et simule le flux de trafic routier de la zone qu'il représente à l'aide d'un modèle qui lui est propre.

3.5.1 Graphe de capteurs

Le découpage du réseau routier n'est pas arbitraire, afin d'éviter les situation incohérentes comme avoir une zone de quelques mètres carrés. Afin de garantir l'intégrité de la simulation, un découpage minimal est déduit à partir des diverses boucles de détection de véhicules [5] : les points d'entrée et de sortie d'un cluster ne peuvent être que des capteurs.

Ainsi, du point de vue du niveau "Contrôle", le réseau routier se résume à un graphe dirigé de capteurs inter-connectés. Chaque arc de ce graphe représente un chemin rejoignant directement deux capteurs, sans passer par un capteur intermédiaire (voir figure 3.12).

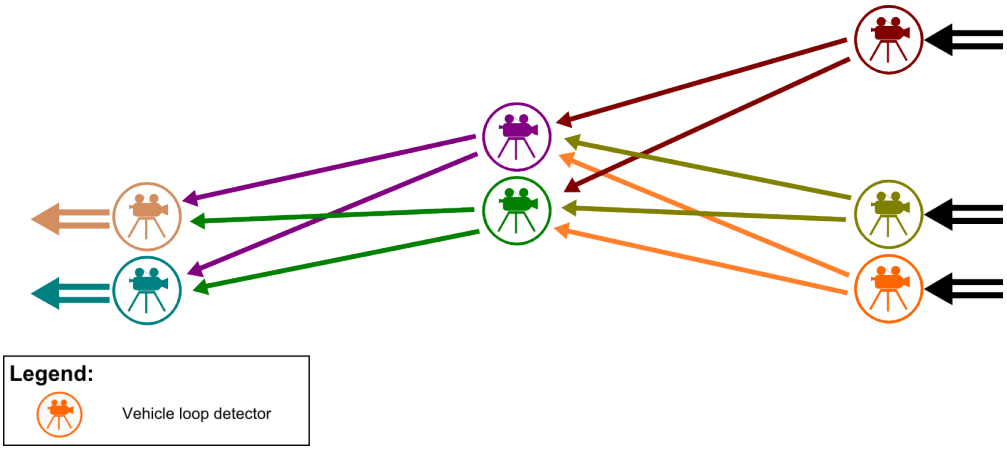


FIGURE 3.12 – Graphe de capteurs de l'exemple présenté dans la figure 3.4.

5. Pour simplifier notre discours, les boucles de détection de véhicules seront appelées *"capteurs"* par la suite

Le bon fonctionnement d'une simulation ne peut être assuré que si les capteurs définis dans le réseau sont cohérents. Pour cela, il faut s'assurer en particulier que :

— Un capteur est présent à chaque borne d'entrée et de sortie du réseau simulé ;
— Si une route a un capteur situé à une entrée, alors toutes les voies de cette route ont un capteur à leur entrée ;
— Si une route a un capteur situé à une sortie, alors toutes les voies de cette route ont un capteur à leur sortie.

3.5.2 Graphe de clusters

Un cluster est ainsi caractérisé par trois ensembles de capteurs :

— Les capteurs situés à leur entrée. Ces capteurs marquent les origines du flux de trafic entrant dans le cluster ;
— Les capteurs situés à leur sortie. Ces capteurs marquent les destinations du flux de trafic sortant du cluster ;
— Les capteurs situés strictement à l'intérieur du cluster.

Le découpage minimal du réseau routier est alors défini par :

— Un cluster représentant l'extérieur du réseau routier. Ce cluster a pour entrées les capteurs situés à la sortie du réseau routier, et a pour sorties les capteurs situés aux entrées du réseau routier ;
— Des cluster minimaux représentant les sous ensembles minimaux du réseau routier. Un cluster est minimal s'il n'y a aucun capteur à l'intérieur du cluster [6].

La figure 3.13 illustre ce découpage minimal dans le cas de l'exemple développé dans ce chapitre.

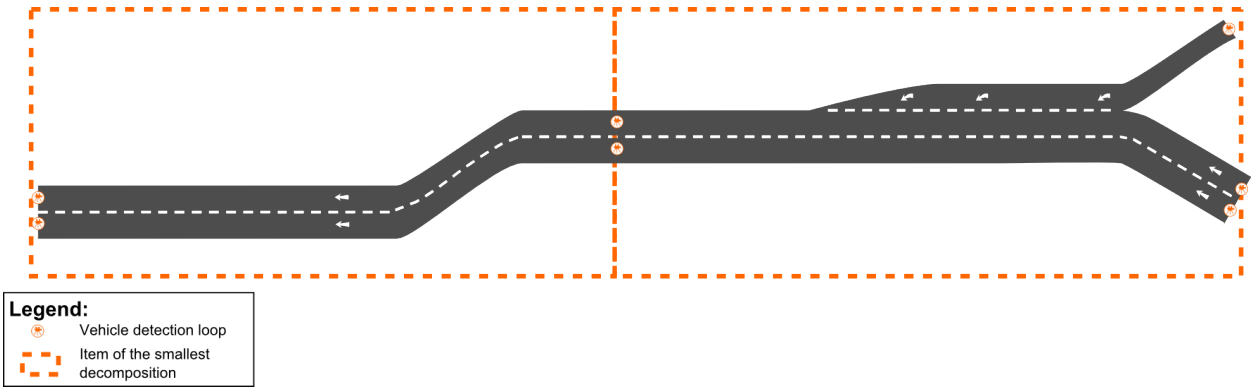


FIGURE 3.13 – Découpage minimal du réseau routier présenté dans la figure 3.4.

Les clusters d'une simulation sont nécessairement une combinaison disjointe de clusters minimaux contigus. Ainsi, dans le cas de l'exemple développé dans ce chapitre, l'ensemble des clusters possibles pour la simulation sont au nombre de quatre [7]. La figure 3.14 résume les décompositions possibles du réseau routier.

6. dans un tel cas, les successeurs des entrées du cluster sont ses sorties, et les antécédents des sorties du cluster sont ses entrées

7. Le cas où tous les clusters sont fusionnés est exclu, étant donné qu'il représente une simulation où tout est extérieur au réseau simulé. Cela correspond donc à une simulation vide.

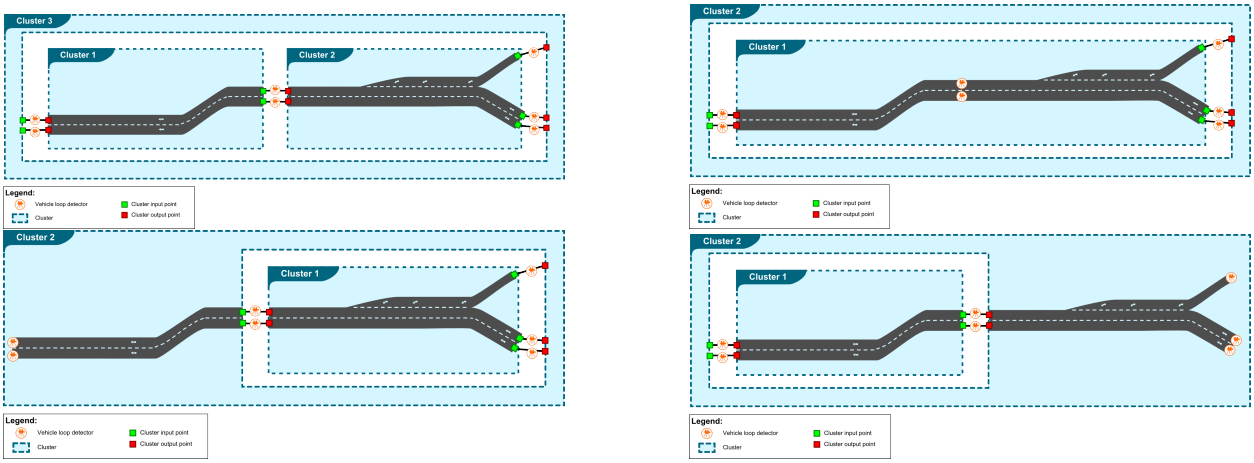


FIGURE 3.14 – Les quatre différentes décompositions possibles en clusters de l'exemple présenté dans la figure 3.4.

3.5.3 Trafic routier dans un cluster

Chaque cluster fonctionne de manière autonome et simule le trafic routier en son sein en utilisant un modèle qui lui est propre. Cette simulation consiste à lire les données présentes aux capteurs à un instant t, afin d'en déduire les valeurs à un instant $t + \Delta t$ (voir figure 3.15) :

— Dans le cas d'un modèle macroscopique, cela consiste à utiliser les formules de conservation de flux et de prédiction pour calculer le flux obtenu à $t + \Delta t$;

— Dans le cas d'un modèle microscopique, cela consiste à modifier la fréquence de génération de véhicules ainsi que la vitesse des véhicules générés en fonction des données des capteurs, ainsi qu'a mesurer le nombre et la vitesse des véhicules passant au niveau des capteurs pour un intervalle de temps donné, afin de déduire les données des capteurs à l'instant $t + \Delta t$.

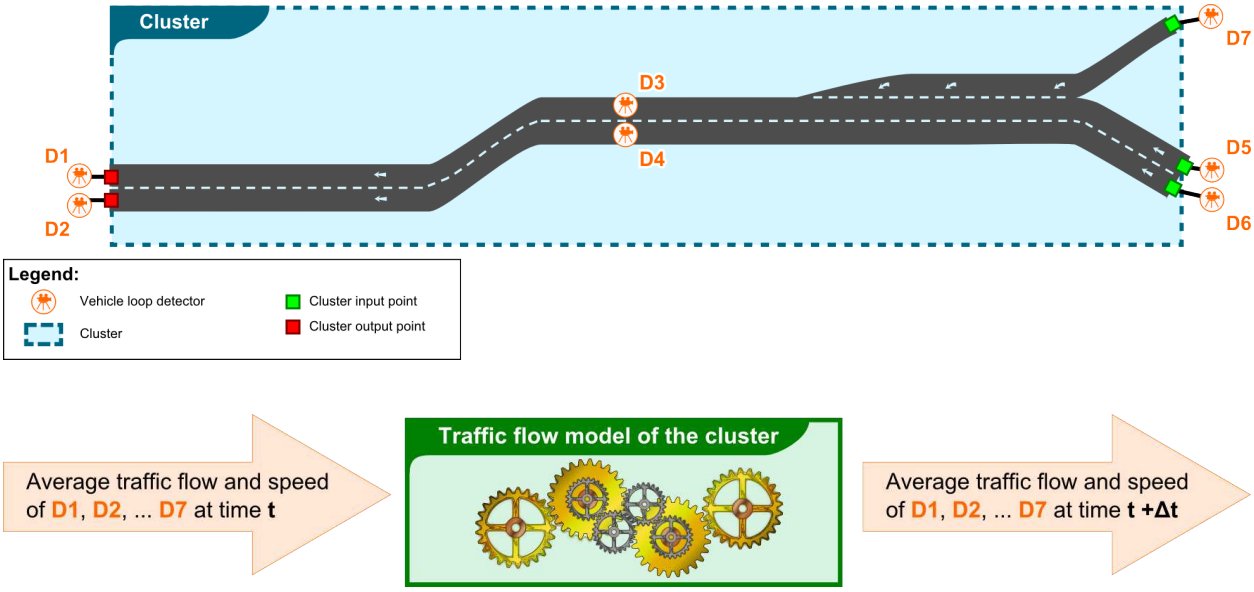


FIGURE 3.15 – Illustration du concept de cluster et de son comportement.

Ainsi, chaque cluster emploie une représentation particulière de la zone du réseau routier qu'il simule, dont la forme dépend du type de modèle utilisé. Un cluster utilise également des agents variant selon les modèles, afin de garantir le bon fonctionnement de la simulation. Par exemple, des agents gérant la génération de véhicules ou le comportement de véhicules dans un modèle micro.

3.5.4 Interactions entre clusters

Chaque cluster effectue de manière quasi-indépendante la simulation du flux de trafic routier sur la partie du réseau qu'il modélise. La seule dépendance existant entre les clusters se situe en leurs points d'entrée et de sortie. En effet, les données qu'ils lisent à un instant t dans les capteurs afin d'en déduire la valeur à l'instant $t + \Delta t$ proviennent des cluster adjacents.

3.5.5 Dynamique des clusters

Lors de l'exécution de la simulation, un agent a pour rôle de superviser la décomposition du réseau en clusters. Cet agent dispose de règles, déterminant si le modèle utilisé dans un cluster doit être modifié (par exemple pour passer de micro à macro) et déterminant si la zone d'un cluster doit être agrandie ou réduite selon les besoins de la simulation (voir l'introduction de ce rapport).

Cet agent peut ainsi décider de retirer un cluster minmal contenu dans un cluster afin de l'ajouter à un autre cluster. Par exemple, dans le cas du suivi d'une congestion présente dans un cluster simulé avec un modèle micro, un cluster minimal pourra être retiré du cluster situé en amont de la congestion afin de l'ajouter au cluster micro.

Cette fonctionnalité dynamique est en cours d'implémentation dans JAM-FREE.

3.6 Modèles de flux de trafic routier

Différents modèles de flux de trafic routier peuvent être utilisés dans JAM-FREE (voir figure 3.16. Actuellement, trois modèles différents sont fournis nativement. Toutefois, son architecture modulaire permet aux utilisateurs à définir leur propre modèle de flux si nécessaire, sous la forme de classes java étendant certaines classes abstraites de JAM-FREE. Les modèles natifs sont :

- Un modèle "Microscopique", où le modèle de poursuite ainsi que le modèle de changement de voie sont par défaut un modèle IDM et un modèle MOBIL. Il est toutefois aisé de substituer à ces modèles un modèle défini par l'utilisateur ;
- Un modèle "Données réelles" produisant aux entrées et sorties les données issues des capteurs réels (par exemples les données de fichiers ODS) ;
- Un modèle "Macroscopique" utilisant un modèle METANET afin de prédire l'évolution du flux de véhicules [8].

L'architecture de JAM-FREE offre même la possibilité de faire de la co-simulation. Cette piste fait partie des perspectives à long terme de JAM-FREE.

3.7 Modèle microscopique

Dans cette section, nous donnons un aperçu du modèle microscopique fourni nativement dans JAM-FREE. Le modèle complet est plus complexe, et a été simplifié afin de faciliter la compréhension des principes régissant notre modèle microscopique.

8. Ce modèle est en cours d'implémentation dans JAM-FREE

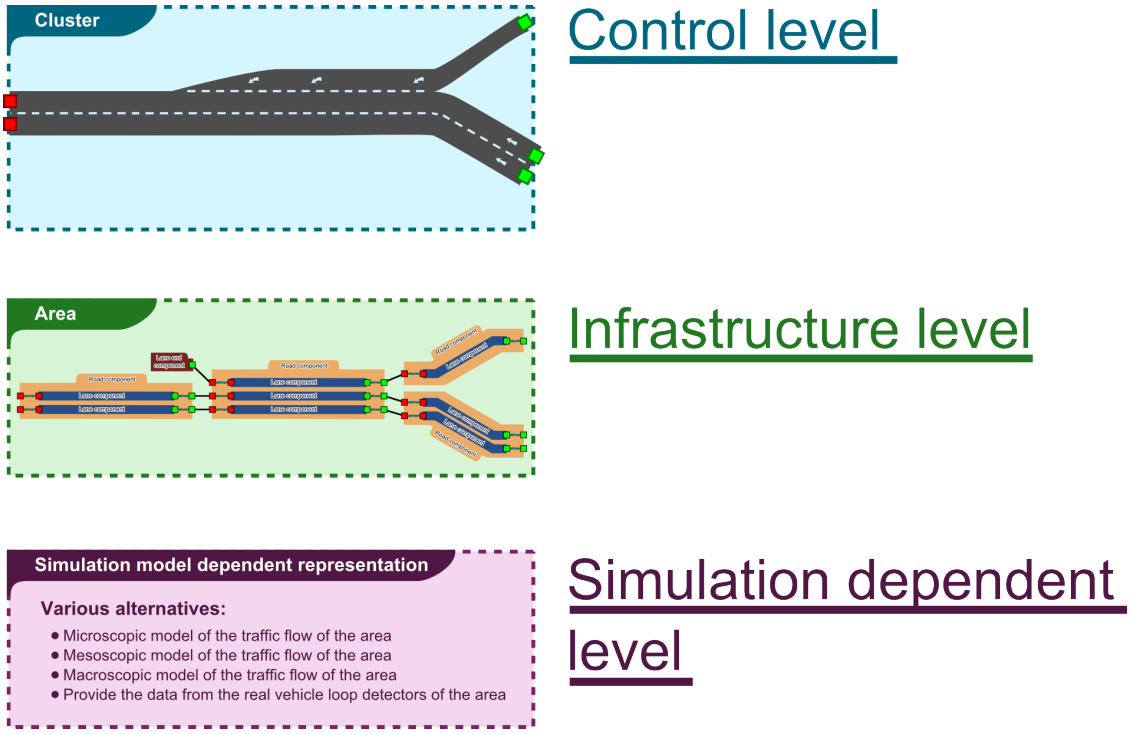


FIGURE 3.16 – Niveaux impliqués dans une simulation hybride dynamique du flux de véhicules.

3.7.1 Réseau routier

Afin de simplifier la tâche de perception des véhicules, ces derniers circulent dans un réseau routier ayant une intelligence ambiante : les voies d'un composant "Route" sont subdivisées en segments, où les règles de circulation sont homogènes (voir figure 3.17). Ainsi, plutôt que de mémoriser tous les panneaux rencontrés afin de déterminer les contraintes de conduites s'imposant à eux, les véhicules n'ont qu'a questionner la segment dans lequel ils se situent.

3.7.2 Véhicules

Chaque véhicule est modélisé par un agent, dont le comportement consiste à déterminer quelle accélération utiliser afin de garantir une vitesse sûre optimale. Son comportement consiste également à identifier à quel moment une manœuvre de changement de voie est intéressante (afin de gagner en vitesse) ou nécessaire (afin de naviguer). Pour cela, l'agent perçoit un certain nombre d'informations concernant sa situation actuelle, afin de décider des actions à effectuer.

Afin de garantir un maximum de liberté et de modularité dans le comportement des véhicules, l'écriture du comportement de chaque véhicule peut être soit intégralement laissé à la charge de l'utilisateur, soit utiliser l'architecture générique proposée par JAM-FREE, où le comportement d'un véhicule est décomposé en trois modules :

— Un module détectant les collisions avec les obstacles proches. Le cas échéant, le véhicule prévient la simulation qu'un accident est en train de se produire ;
— Un module de suivi de véhicules, calculant l'accélération du véhicule en fonction des données qu'il a perçues ;
— Un module de changement de voie, déterminant quand un changement de voie est sûr, intéressant ou nécessaire. Le cas échéant, ce module initie le changement de voie.

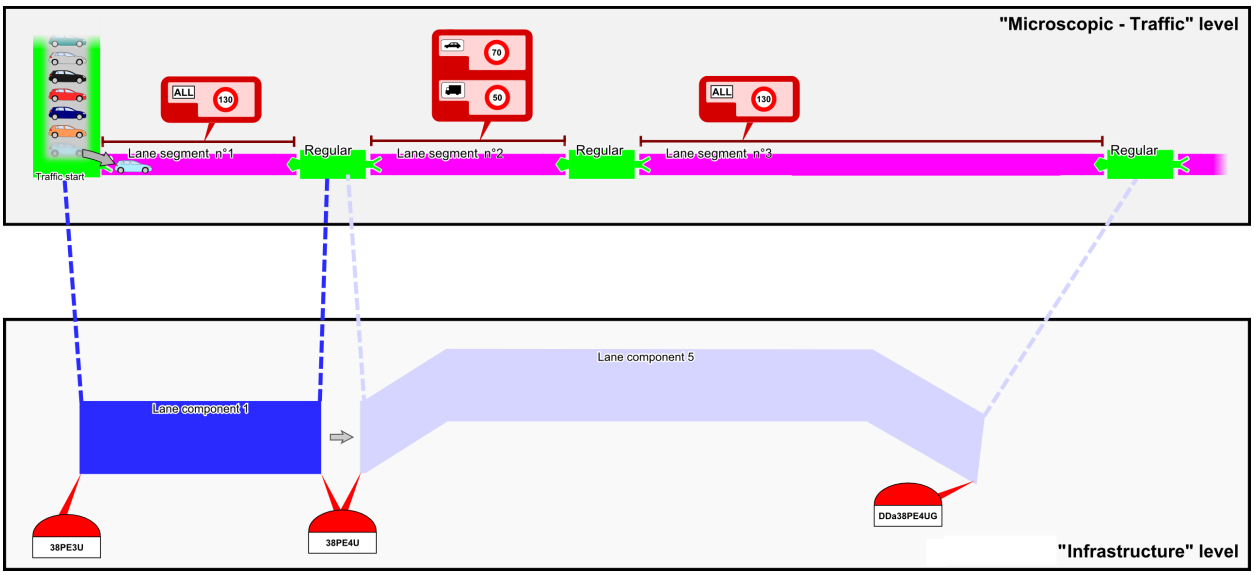


FIGURE 3.17 – Modèle d'une voie au niveau "Microscopique"

Perception

Chaque véhicule dispose de son propre modèle de perception paramétrable. Par défaut, ce modèle de perception récupère les informations présentes dans le modèle du réseau routier, afin de construire une représentation centrée sur le véhicule. Ce modèle relatif fournit les informations suivantes concernant la voie courante du véhicule, la voie située sur sa gauche et la voie située sur sa droite (si ces voies existent) :

- les contraintes de circulation s'appliquant au véhicule dans le segment actuel de cette voie ;
- l'obstacle frontal le plus proche, la distance entre le véhicule et cet obstacle ainsi que diverses propriétés de l'obstacle (comme sa vitesse) ;
- l'obstacle arrière le plus proche, la distance entre le véhicule et cet obstacle ainsi que diverses propriétés de l'obstacle (comme sa vitesse).

Ce modèle relatif est illustré par la figure 3.18.

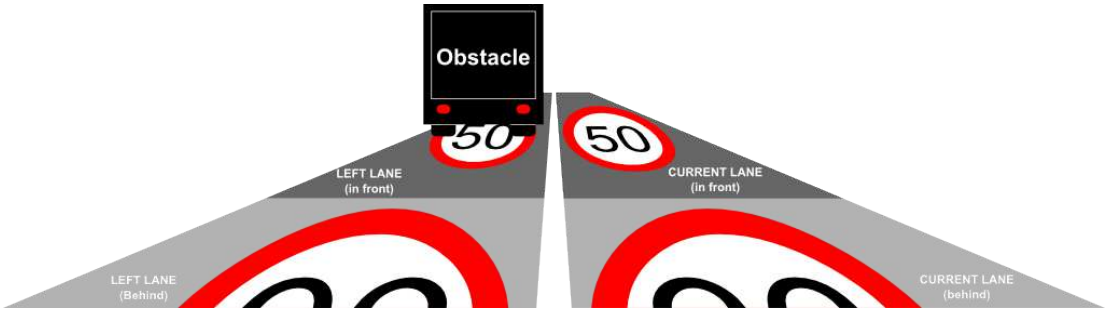


FIGURE 3.18 – Données perçues par un véhicule afin d'effectuer des décisions.

Suivi de véhicules

JAM-FREE implémente par défaut le modèle IDM (Intelligent-Driver Model) pour simuler le comportement d'accélération des véhicules (Kesting *et al.*, 2010). Toutefois, l'utilisation de ce

modèle n'est pas impératif : il est tout à fait possible de définir son propre modèle d'accélération.

IDM est un modèle de flux de trafic microscopique, où la décision d'un conducteur d'accélérer ou de freiner ne dépend que de sa propre vitesse et de la position et de la vitesse du véhicule situé immédiatement en avant. Un changement de voie, cependant, dépendra de tous les véhicules voisins (voir le modèle de changement de voie MOBIL).

Changement de voie

Les changements de voies ont lieu si :

— une autre voie est plus attrayante (critère d'incitation),

— et la modification peut être effectuée en toute sécurité (critère de sécurité).

Dans JAM-FREE, le modèle de changement de voie utilisé par défaut est une version très fortement inspirée du modèle de changement de voie MOBIL [9].

Ces différents modèles sont décrits précisément dans la documentation de JAM-FREE.

3.7.3 Capteurs

Afin de mettre à jour les données des capteurs, un agent est associé à chaque capteur du niveau "Microscopique". Ces agents ont pour rôle de mesurer le flux moyen et la vitesse moyenne des véhicules passant par le capteur pendant un intervalle de temps donné, afin de calculer le flux et la vitesse moyenne mesurée par le capteur au temps $t + \Delta t$.

3.7.4 Points de génération de véhicules

Afin de créer les véhicules circulant au niveau "Microscopique", des agents sont créés à chaque capteur d'entrée du cluster. Ces agents ont pour rôle de comparer la valeur du flux fourni en entrée par le cluster en amont au flux mesuré par le capteur présent dans le modèle "Microscopique". En fonction de la divergence de ces deux valeurs, l'agent peut décider de générer un véhicule roulant à la vitesse moyenne lue (si c'est possible).

Le modèle de génération de véhicules utilisé par défaut génère des véhicules fonctionnant avec un modèle IDM combiné avec un modèle MOBIL. Toutefois, les utilisateurs sont libres de changer ce modèle pour un autre, s'ils le considèrent comme plus approprié.

3.7.5 Interfaces utilisateur

Des modèles et simulations JAM-FREE peuvent être facilement créés à l'aide d'un système hybride XML/Java :

— les simulations utilisant les modèles comportementaux prédéfinis (IDM et MOBIL modifiés) peuvent être créées en utilisant seulement des fichiers de paramétrage au format XML,

— les simulations plus complexes peuvent être créées en deux étapes : 1) la création de classes Java décrivant les nouveau comportements et 2) la mention de ces nouvelles classes dans les fichiers de paramétrage au formal XML.

Afin de faciliter la conception de simulations portant sur un même réseau routier, un modèle comprend plusieurs fichiers XML :

— un fichier principal décrivant la simulation ;

— un fichier décrivant le réseau routier ;

— un fichier décrivant le code de la route (*i.e.* les règles de circulation par défaut) ;

9. `http://xxx.uni-augsburg.de/abs/cond-mat/0002177`

— un fichier identifiant la position des divers capteurs présents dans le réseau, ainsi que les données réelles leur étant associées ;
— un fichier de données réelles de flux de trafic par capteur ;
— un fichier décrivant la position et la nature des panneaux de signalisation présents dans le réseau ;
— un fichier identifiant le découpage initial du réseau en clusters, ainsi que le modèle utilisé initialement dans les clusters ;
— un fichier par modèle de flux de trafic routier, définissant les paramètres et le calibrage utilisés par le modèle (*e.g.* la valeur des paramètres a, b, s_0 et autres du modèle IDM utilisé au niveau micro).

Le fichier principal de la simulation définit en particulier le générateur de nombres aléatoires, le modèle de temps utilisé, la précision des calculs effectués, les différents modèles de flux de trafic de pouvant être utilisés dans la simulation et le nom des fichiers mentionnés par les points de cette énumération. Un exemple de simulation est illustré à l'annexe D page 37.

Pour gérer simplement des simulations, une interface utilisateur graphique appelée JFF (JAM-FREE Framework) a également été développée (voir l'annexe E page 45).

Chapitre 4

Conclusion

Dans le cadre du projet ISART de CISIT nous développons un simulateur hybride dynamique de flux de trafic appelé JAM-FREE, reposant sur le méta-modèle SIMILAR. Ces logiciels sont développés selon les meilleures pratiques de génie logiciel et à l'aide de nombreux outils comme SonarQube [1] (voir l'annexe F), Maven [2], Jenkins [3] ou Hamcrest [4]. Des tests unitaires exhaustifs ont été effectués pour vérifier les moteurs de simulation et les modèles. Par exemple, 545800 tests (265Mo de code source), chacun décrivant un cas d'utilisation spécifique, ont été générés pour valider le modèle de perception de JAM-FREE.

Délivrables

Le méta-modèle et le moteur de simulation de SIMILAR ont été implémentés en Java, et sont disponibles librement sous licence CeCILL-B [5]. Les sources du projet sont pour le moment hébergées sur `https://forge.univ-artois.fr` avec un accès public au code. Les binaires et la documentations sont disponibles sur le site personnel de Gildas Morvan [6]

JAM-FREE est encore en développement. Cependant, des démonstrations préliminaires peuvent être distribuées aux partenaires de CISIT si nécessaire.

Deux articles sont en préparation :

— Morvan et Kubera (2014b), décrivant le méta-modèle SIMILAR,

— Morvan et Kubera (2014a), proposant de nouvelles façons de gérer l'incertitude dans les simulations multi-agents en utilisant la théorie des fonctions de croyance.

1. `http://www.sonarqube.org`
2. `http://maven.apache.org`
3. `jenkins-ci.org`
4. `https://code.google.com/p/hamcrest/`
5. `http://cecill.info/licences/Licence_CeCILL-B_V1-en.html`
6. `http://www.lgi2a.univ-artois.fr/~morvan/similar.html`

Annexe A

Images agrandies

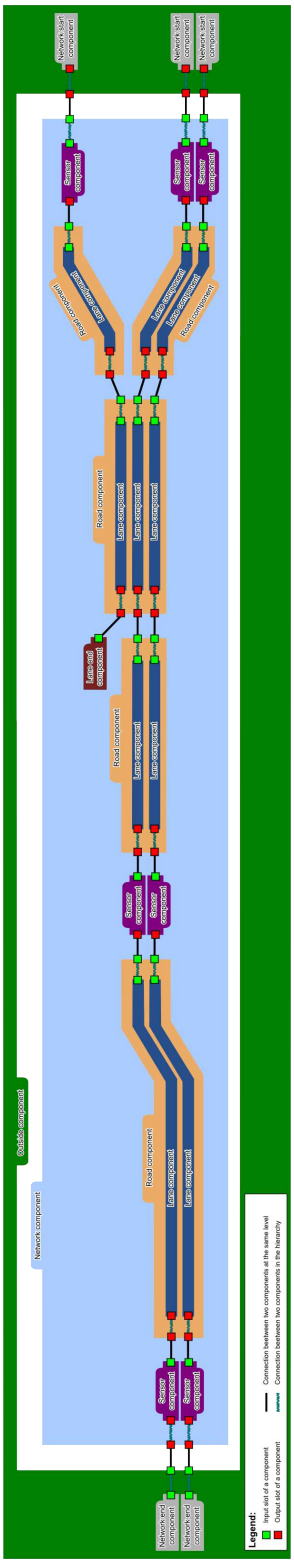


FIGURE A.1 – Illustration d'un modèle de réseau routier correspondant à celui de la figure 3.4, du point de vue du niveau "Infrastructure". Cette image est une version agrandie de la figure 3.7 page 17

Annexe B

Illustration des cas d'utilisation définis dans JAM-FREE (anglais)

B.1 Reduce the CPU load during peak hours

The hybrid model can change the traffic representation of a portion of the road network dynamically. This feature can be used to switch from a microscopic representation to a macroscopic representation when the CPU is overused in order to reduce its load. Such situations include cases where the number of simulated vehicles becomes too high to be managed satisfyingly by the CPU (for instance during peak hours). This case is illustrated in figure B.1.

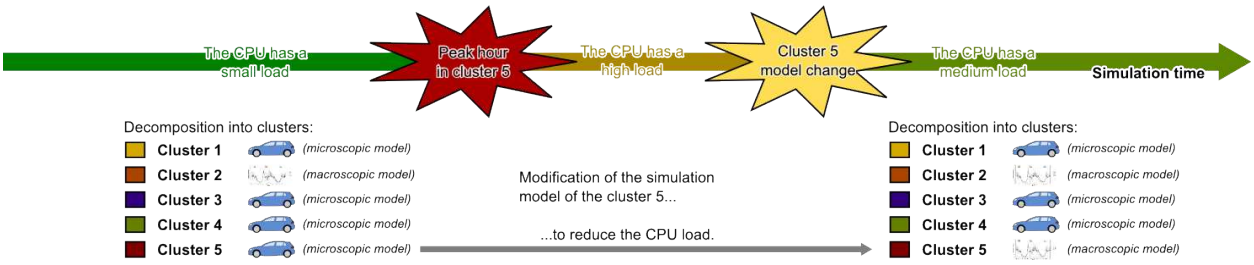


FIGURE B.1 – Illustration of the use case "Reduce the CPU load during peak hours".

B.2 Increase the details on the traffic during off-peak hours

The hybrid model can change the traffic representation of a portion of the road network dynamically. This feature can be used to switch from a macroscopic representation to a microscopic representation when the CPU is underused in order to increase the details on the traffic. Such situations include cases where the number of simulated vehicles is significantly reduced (for instance during off-peak hours). This case is illustrated in figure B.2.

B.3 Balance the CPU load between clusters

The hybrid model can change the traffic representation of a portion of the road network dynamically. This feature can be used to switch from :

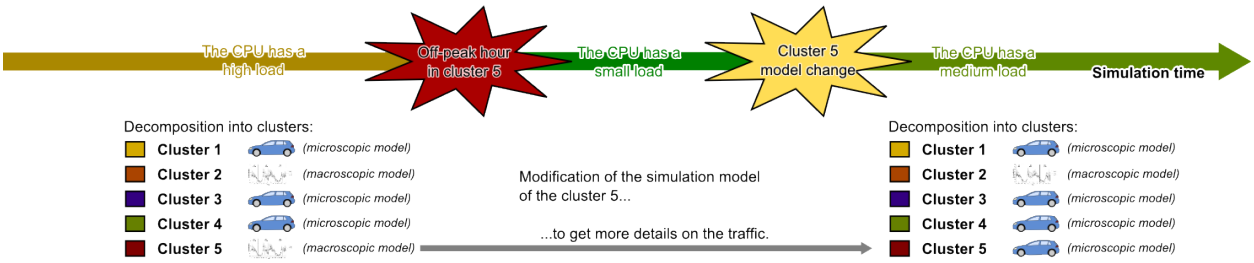


FIGURE B.2 – Illustration of the use case "Increase the details on the traffic during off-peak hours".

— a microscopic representation to a macroscopic representation when the CPU is overused in order to reduce its load ;
— a macroscopic representation to a microscopic representation to find out the reason why the traffic jam appeared.

These features can be used jointly to balance the load between two clusters.

Such situations include cases where a traffic jam appears in a cluster using a macroscopic model and where the number of simulated vehicles in another cluster is significantly reduced (for instance during off-peak hours). This case is illustrated in figure B.3.

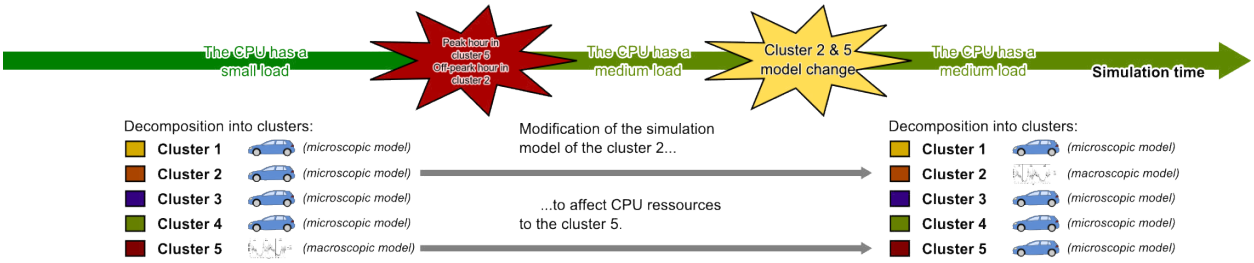


FIGURE B.3 – Illustration of the use case "Balance the CPU load between clusters".

Such situations also include cases where a traffic jam appears in a cluster using a microscopic model and where the number of simulated vehicles in another cluster is significantly reduced (for instance during off-peak hours). This case is illustrated in figure B.4.

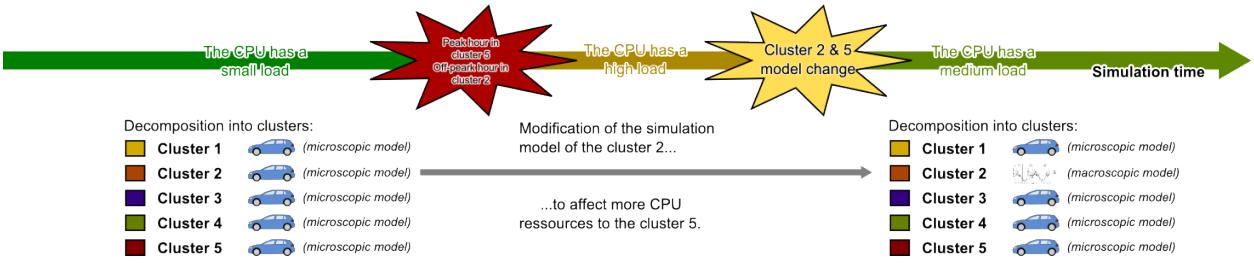


FIGURE B.4 – Illustration of the use case "Balance the CPU load between clusters".

B.4 Identify and locate traffic jams

The clusters used in the hybrid model are not static. Their number can be increased by divinding existing clusters into two or more clusters. This feature can be used to locate traffic jams. Indeed,

macroscopic representations provide quantitative information concerning the traffic, including the mean speed and the free driving speed. A significant difference between these values indicates the presence of a traffic jam. Finding the area where the traffic jam is located consists in dividing the clusters and focusing on the ones where the mean speed is significantly lower than the free driving speed. This case is illustrated in figure B.5.

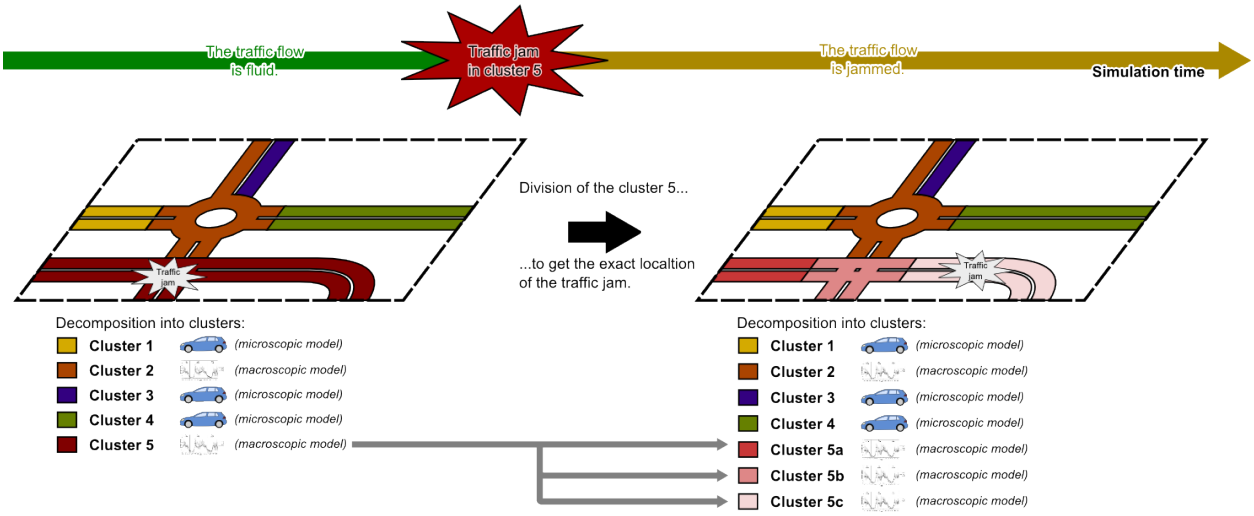


FIGURE B.5 – Illustration of the use case "Identify and locate traffic jams".

B.5 Find out the reason of a traffic jam

The hybrid model can change the traffic representation of a portion of the road network dynamically. This feature can be used to switch from a macroscopic representation to a microscopic representation to find out the reason why the traffic jam appeared. This case is illustrated in figure B.6.

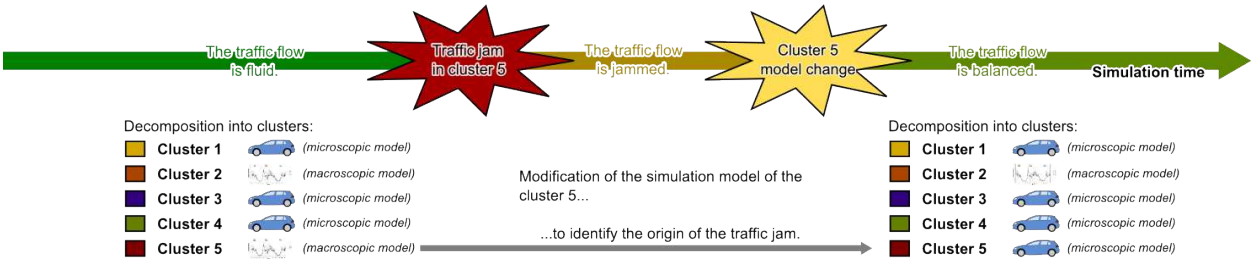


FIGURE B.6 – Illustration of the use case "Find out the reason of a traffic jam".

B.6 Follow moving macroscopic phenomena

The static division of the road network into clusters that can be found in most existing approaches causes precision losses. Indeed, a traffic flow simulator has to be able to reproduce macroscopic phenomena such as shockwaves (see figure B.7).

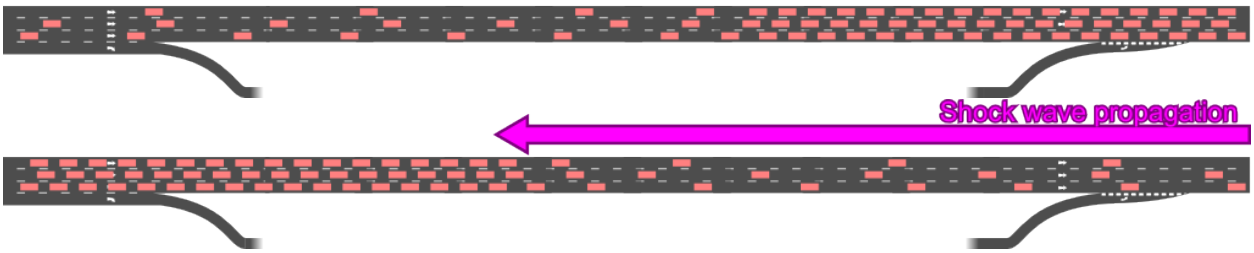


FIGURE B.7 – Illustration of a shock wave phenomenon.

Yet, the static decomposition found in static hybrid models will cause eventually a loss a quantitative data (see figure B.8).

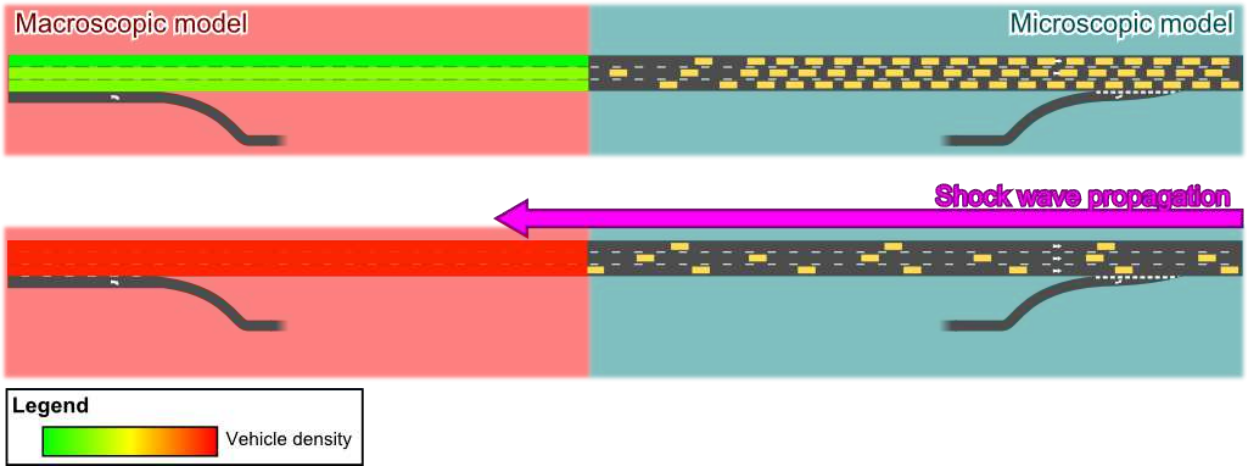


FIGURE B.8 – Static hybrid model of a shockwave phenomenon.

To avoid this issue, JAM-FREE gives the ability to move, resize, split and merge to the clusters of the road network (see figure B.9).

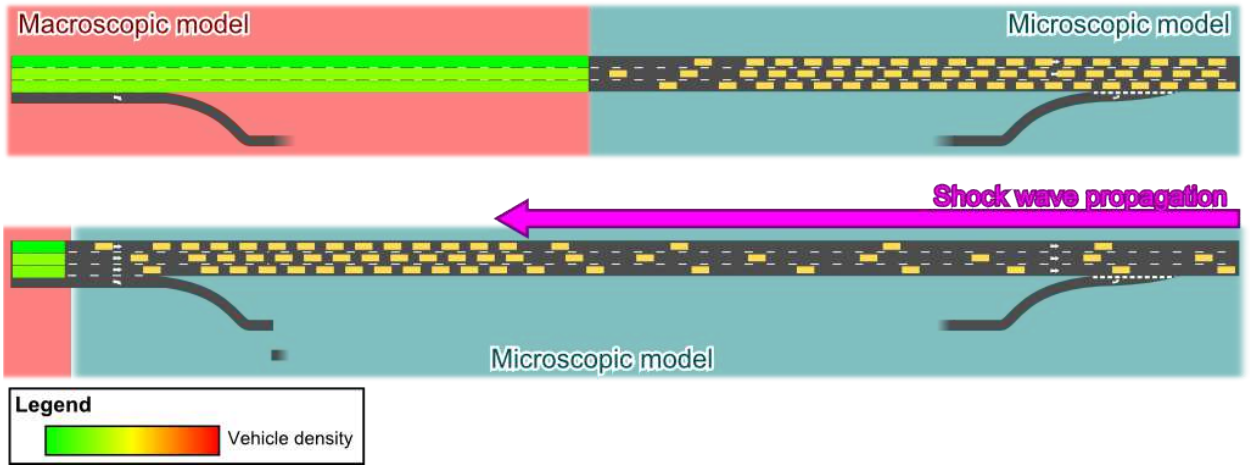
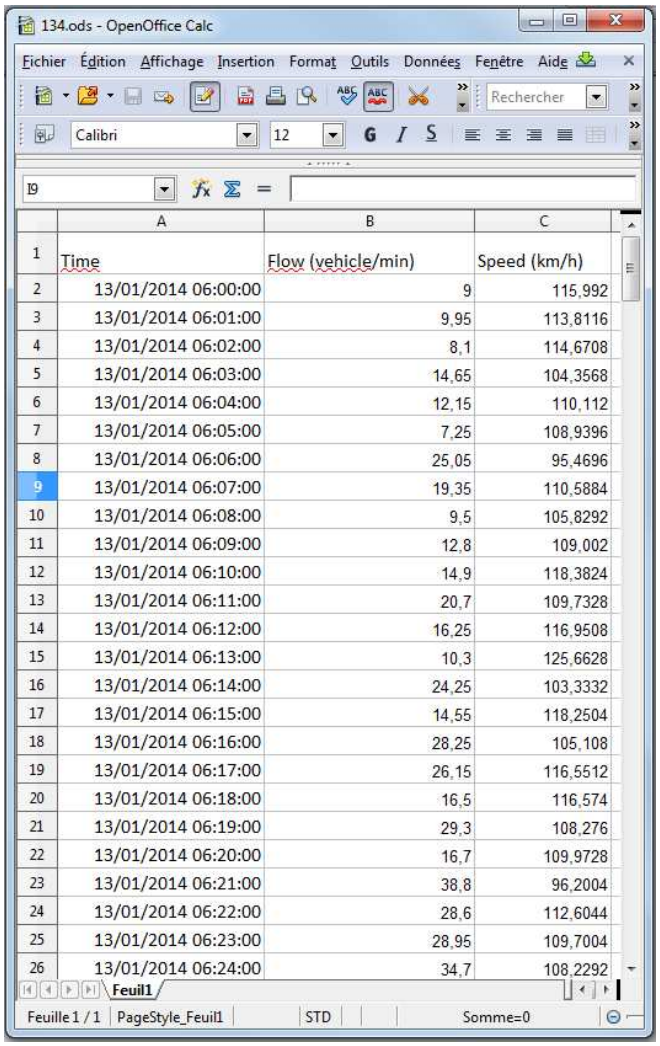


FIGURE B.9 – Dynamic hybrid model of a shockwave phenomenon.

Annexe C

Données réelles associées à une boucle de détection de véhicules



134.ods - OpenOffice Calc

Fichier Édition Affichage Insertion Format Outils Données Fenêtre Aide

Rechercher

Calibri 12

	A	B	C
1	Time	Flow (vehicle/min)	Speed (km/h)
2	13/01/2014 06:00:00	9	115,992
3	13/01/2014 06:01:00	9,95	113,8116
4	13/01/2014 06:02:00	8,1	114,6708
5	13/01/2014 06:03:00	14,65	104,3568
6	13/01/2014 06:04:00	12,15	110,112
7	13/01/2014 06:05:00	7,25	108,9396
8	13/01/2014 06:06:00	25,05	95,4696
9	13/01/2014 06:07:00	19,35	110,5884
10	13/01/2014 06:08:00	9,5	105,8292
11	13/01/2014 06:09:00	12,8	109,002
12	13/01/2014 06:10:00	14,9	118,3824
13	13/01/2014 06:11:00	20,7	109,7328
14	13/01/2014 06:12:00	16,25	116,9508
15	13/01/2014 06:13:00	10,3	125,6628
16	13/01/2014 06:14:00	24,25	103,3332
17	13/01/2014 06:15:00	14,55	118,2504
18	13/01/2014 06:16:00	28,25	105,108
19	13/01/2014 06:17:00	26,15	116,5512
20	13/01/2014 06:18:00	16,5	116,574
21	13/01/2014 06:19:00	29,3	108,276
22	13/01/2014 06:20:00	16,7	109,9728
23	13/01/2014 06:21:00	38,8	96,2004
24	13/01/2014 06:22:00	28,6	112,6044
25	13/01/2014 06:23:00	28,95	109,7004
26	13/01/2014 06:24:00	34,7	108,2292

Feuil1

Feuille 1 / 1 PageStyle_Feuil1 STD Somme=0

FIGURE C.1 – Exemple d'un fichier ODS contenant des données réelles d'une boucle de détection de véhicules.

Annexe D

Fichiers XML de JAM-FREE

Ce chapitre décrit comment les fichiers XML sont utilisés afin de créer une simulation d'une portion de l'autoroute A25 à la hauteur de la Chapelle d'Armentières, en utilisant un unique cluster microscopique pour simuler le flux de trafic routier dans le réseau. Le réseau utilisé dans cet exemple contient une voie d'insertion et une voie d'extraction (voir figure D.1).

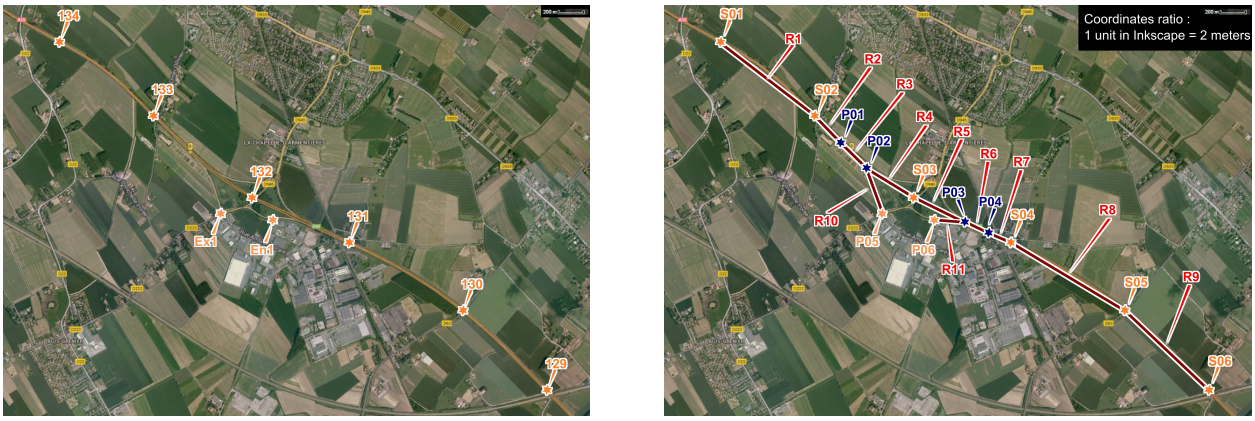


FIGURE D.1 – Vue aérienne du réseau simulé. Les capteurs y sont identifiés par des points oranges (gauche). Les différents composants route du modèle du réseau y sont également représentés (droite).

D.1 Liste des fichiers

Dans JAM-FREE, un modèle comprend plusieurs fichiers XML :
- le fichier principal *"simulationModel.xml"* décrivant la simulation ;
- le fichier *"infrastructure.xml"* décrivant le réseau routier ;
- le fichier *"drivingRules.xml"* décrivant le code de la route ;
- le fichier *"flowSensors.xml"* modélisant les capteurs de la simulation ;
- les fichiers *"129.ods"*, *"130.ods"*, *"132.ods"* et *"134.ods"* contenant les données réelles de flux de trafic par capteur [1] ;
- le fichier *"trafficSigns.xml"* décrivant la position et la nature des panneaux de signalisation présents dans le réseau ;
- le fichier *"initialClusters.xml"* identifiant le découpage initial du réseau en clusters ;

1. ces fichiers ne sont pas inclus dans cet annexe

— les fichiers *"microWithSimpleIDMAndMobil.xml"* et *"realData.xml"* décrivant les paramètres des modèle de flux de trafic routier.

D.2 Fichier principal

Le fichier principal de la simulation définit en particulier le générateur de nombres aléatoires, le modèle de temps utilisé, la précision des calculs effectués, les différents modèles de flux de trafic de pouvant être utilisés dans la simulation et le nom des autres fichiers définissant le modèle.

Listing D.1 – JAM-FREE : main XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<XMLSimulationDescription>
<!--                          -->
<!-- Definition of the random seed, initializing the random number generator.        -->
<!--                          -->
       <randomSeed value="1" />

<!--                          -->
<!-- Definition of the items initializing the state of the environment of the simulation.    -->
<!--                          -->
       <levelsInitializers>
              <className>fr.lgi2a.jamfree.flowmodel.microscopic.MicroscopicLevelInitializer</className>
              <className>fr.lgi2a.jamfree.flowmodel.realdata.RealDataLevelInitializer</className>
       </levelsInitializers>

<!--                          -->
<!-- Definition of the time model of the simulation.             -->
<!--                          -->
       <timeModel>
              <timeUnit    seconds="0.01" />
              <simulationStart  date="2014-01-13T06:00:00" />
              <simulationEnd   date="2014-01-13T10:00:00" />
              <levelsTimeModel>
                     <timeModel   level="control"    timeUnitsPeriod="40" />
                     <timeModel   level="infrastructure"   timeUnitsPeriod="10" />
                     <timeModel   level="microscopic"   timeUnitsPeriod="1" />
                     <timeModel   level="real data"    timeUnitsPeriod="6000" />
              </levelsTimeModel>
       </timeModel>
<!--                          -->
<!-- Definition of the computations precision model of the simulation.          -->
<!--                          -->
       <precisionModels>
              <precisionModel  valueType="distance"   precision="0.001" />
              <precisionModel  valueType="duration"  precision="0.001" />
              <precisionModel  valueType="weight"    precision="0.001" />
              <precisionModel  valueType="height"    precision="0.001" />
              <precisionModel  valueType="speed"    precision="0.001" />
              <precisionModel  valueType="acceleration"  precision="0.001" />
       </precisionModels>
<!--                          -->
<!-- Definition of the road network of the simulation.             -->
<!--                          -->
       <networkModel>
              <defaultDrivingRules  file="drivingRules.xml" />
              <infrastructure   file="infrastructure.xml" />
              <flowsensors    file="flowSensors.xml" />
              <trafficSigns    file="trafficSigns.xml" />
       </networkModel>
<!--                          -->
<!-- Definition of the various models that can be used to simulate the traffic flow.      -->
<!--                          -->
       <trafficFlowModels>
              <trafficFlowModel   alias="Micro model with Simple IDM and MOBIL"
                                                descriptorFile="trafficflowmodels/microWithSimpleIDMAndMobil.xml"
              />
              <trafficFlowModel   alias="Real data"
                                                descriptorFile="trafficflowmodels/realData.xml"
              />
       </trafficFlowModels>
<!--                          -->
<!-- Definition of the file describing the initial decomposition of the simulation as clusters.   -->
<!--                          -->
       <initialClusters    file="initialClusters.xml" />
</XMLSimulationDescription>
```

D.3 Modèle du réseau routier

Le modèle du niveau ”Infrastructure” du réseau routier est décrit dans les deux fichiers qui suivent.

Listing D.2 – JAM-FREE : infrastructure

```
<?xml version="1.0" encoding="UTF-8"?>
<infrastructure>
<!--
	Description of the road components contained in the road network.
-->
	<road id="R1" numberOfLanes="2" width="4" semanticLength="960">
		<type value="highway"/>
		<input plo="A0025PR017" distanceFromPlo="650">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR016" distanceFromPlo="690">
			<laneIdRange from="0" to="1"/>
		</output>
		<shape>
			<point x="438.892" y="2801.16"/>
			<point x="1072.078" y="2178.704"/>
		</shape>
	</road>
	<road id="R2" numberOfLanes="2" width="4" semanticLength="297">
		<type value="highway"/>
		<input plo="A0025PR016" distanceFromPlo="690">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR016" distanceFromPlo="413">
			<laneIdRange from="0" to="1"/>
		</output>
		<shape>
			<point x="1072.078" y="2178.704"/>
			<point x="1263.89" y="1979.442"/>
		</shape>
	</road>
	<road id="R3" numberOfLanes="3" width="4" semanticLength="288">
		<type value="highway"/>
		<input plo="A0025PR016" distanceFromPlo="413">
			<laneIdRange from="-1" to="1"/>
		</input>
		<output plo="A0025PR016" distanceFromPlo="145">
			<laneIdRange from="-1" to="1"/>
		</output>
		<characterization>
			<laneStartingPoint inputId="-1" />
		</characterization>
		<shape>
			<point x="1261.0892887419732" y="1976.5861259745636"/>
			<point x="1452.5412887419732" y="1788.8321259745637"/>
		</shape>
	</road>
	<road id="R4" numberOfLanes="2" width="4" semanticLength="425">
		<type value="highway"/>
		<input plo="A0025PR016" distanceFromPlo="145">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR015" distanceFromPlo="680">
			<laneIdRange from="0" to="1"/>
		</output>
		<shape>
			<point x="1455.342" y="1791.688"/>
			<point x="1802.006" y="1571.812"/>
		</shape>
	</road>
	<road id="R5" numberOfLanes="2" width="4" semanticLength="442">
		<type value="highway"/>
		<input plo="A0025PR015" distanceFromPlo="680">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR015" distanceFromPlo="260">
			<laneIdRange from="0" to="1"/>
		</output>
		<shape>
			<point x="1802.006" y="1571.812"/>
			<point x="2181.312" y="1393.84"/>
		</shape>
	</road>
	<road id="R6" numberOfLanes="3" width="4" semanticLength="216">
		<type value="highway"/>
		<input plo="A0025PR015" distanceFromPlo="260">
			<laneIdRange from="-1" to="1"/>
		</input>
		<output plo="A0025PR015" distanceFromPlo="66">
			<laneIdRange from="-1" to="1"/>
		</output>
		<characterization>
```

```
                <laneEndingPoint outputId="-1" />
            </characterization>
            <shape>
                <point x="2179.6503504762213" y="1390.2014672105192"/>
                <point x="2355.7863504762213" y="1309.7634672105194"/>
            </shape>
        </road>
        <road id="R7" numberOfLanes="2" width="4" semanticLength="222">
            <type value="highway"/>
            <input plo="A0025PR015" distanceFromPlo="66">
                <laneIdRange from="0" to="1"/>
            </input>
            <output plo="A0025PR14" distanceFromPlo="830">
                <laneIdRange from="0" to="1"/>
            </output>
            <shape>
                <point x="2357.448" y="1313.402"/>
                <point x="2520.86" y="1240.464"/>
            </shape>
        </road>
        <road id="R8" numberOfLanes="2" width="4" semanticLength="1040">
            <type value="highway"/>
            <input plo="A0025PR14" distanceFromPlo="830">
                <laneIdRange from="0" to="1"/>
            </input>
            <output plo="A0025PR13" distanceFromPlo="790">
                <laneIdRange from="0" to="1"/>
            </output>
            <shape>
                <point x="2520.86" y="1240.464"/>
                <point x="3363.342" y="738.21"/>
            </shape>
        </road>
        <road id="R9" numberOfLanes="2" width="4" semanticLength="920">
            <type value="highway"/>
            <input plo="A0025PR13" distanceFromPlo="790">
                <laneIdRange from="0" to="1"/>
            </input>
            <output plo="A0025PR12" distanceFromPlo="870">
                <laneIdRange from="0" to="1"/>
            </output>
            <shape>
                <point x="3363.342" y="738.21"/>
                <point x="3985.722" y="148.758"/>
            </shape>
        </road>
        <road id="R10" numberOfLanes="1" width="4" semanticLength="356">
            <type value="highway"/>
            <input plo="A0025PR016" distanceFromPlo="145">
                <laneId id="-1"/>
            </input>
            <output plo="A0025PR016" distanceFromPlo="501">
                <laneId id="-1"/>
            </output>
            <shape>
                <point x="1452.5412887419732" y="1788.8321259745637"/>
                <point x="1569.698" y="1453.57"/>
            </shape>
        </road>
        <road id="R11" numberOfLanes="1" width="4" semanticLength="226">
            <type value="highway"/>
            <input plo="A0025PR015" distanceFromPlo="34">
                <laneId id="-1"/>
            </input>
            <output plo="A0025PR015" distanceFromPlo="260">
                <laneId id="-1"/>
            </output>
            <shape>
                <point x="1955.108" y="1406.484"/>
                <point x="2179.6503504762213" y="1390.2014672105192"/>
            </shape>
        </road>

<!--
        Description of the connections between the roads.
-->
        <connection outputComponentId="R1" inputComponentId="R2">
            <input plo="A0025PR016" distanceFromPlo="690">
                <laneIdRange from="0" to="1"/>
            </input>
            <output plo="A0025PR016" distanceFromPlo="690">
                <laneIdRange from="0" to="1"/>
            </output>
        </connection>
        <connection outputComponentId="R2" inputComponentId="R3">
            <input plo="A0025PR016" distanceFromPlo="413">
                <laneIdRange from="0" to="1"/>
            </input>
            <output plo="A0025PR016" distanceFromPlo="413">
                <laneIdRange from="0" to="1"/>
            </output>
        </connection>
```

```
	<connection outputComponentId="R3" inputComponentId="R4">
		<input plo="A0025PR016" distanceFromPlo="145">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR016" distanceFromPlo="145">
			<laneIdRange from="0" to="1"/>
		</output>
	</connection>
	<connection outputComponentId="R4" inputComponentId="R5">
		<input plo="A0025PR015" distanceFromPlo="680">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR015" distanceFromPlo="680">
			<laneIdRange from="0" to="1"/>
		</output>
	</connection>
	<connection outputComponentId="R5" inputComponentId="R6">
		<input plo="A0025PR015" distanceFromPlo="260">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR015" distanceFromPlo="260">
			<laneIdRange from="0" to="1"/>
		</output>
	</connection>
	<connection outputComponentId="R6" inputComponentId="R7">
		<input plo="A0025PR015" distanceFromPlo="66">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR015" distanceFromPlo="66">
			<laneIdRange from="0" to="1"/>
		</output>
	</connection>
	<connection outputComponentId="R7" inputComponentId="R8">
		<input plo="A0025PR14" distanceFromPlo="830">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR14" distanceFromPlo="830">
			<laneIdRange from="0" to="1"/>
		</output>
	</connection>
	<connection outputComponentId="R8" inputComponentId="R9">
		<input plo="A0025PR13" distanceFromPlo="790">
			<laneIdRange from="0" to="1"/>
		</input>
		<output plo="A0025PR13" distanceFromPlo="790">
			<laneIdRange from="0" to="1"/>
		</output>
	</connection>

	<connection outputComponentId="R3" inputComponentId="R10">
		<input plo="A0025PR016" distanceFromPlo="145">
			<laneId id="-1"/>
		</input>
		<output plo="A0025PR016" distanceFromPlo="145">
			<laneId id="-1"/>
		</output>
	</connection>
	<connection outputComponentId="R11" inputComponentId="R6">
		<input plo="A0025PR015" distanceFromPlo="260">
			<laneId id="-1"/>
		</input>
		<output plo="A0025PR015" distanceFromPlo="260">
			<laneId id="-1"/>
		</output>
	</connection>

</infrastructure>
```

Listing D.3 – JAM-FREE : traffic signs

```
<?xml version="1.0" encoding="UTF-8"?>
<XMLNetworkSignsDescription>
</XMLNetworkSignsDescription>
```

D.4 Code de la route

Le modèle XML du code de la route définit les contraintes de conduite par défaut en l'absence de signalisation et panneaux.

Listing D.4 – JAM-FREE : driving rules

```
<?xml version="1.0" encoding="UTF-8"?>
<XMLDefaultRulesDescription>
	<highway>
```

```
		<defaultConstraints>
			<maxSpeed value="36.11112"/>
		</defaultConstraints>
	</highway>
	<national>
		<defaultConstraints>
			<maxSpeed value="25"/>
		</defaultConstraints>
	</national>
	<nationalWithMedianStrip>
		<defaultConstraints>
			<maxSpeed value="30.55556"/>
		</defaultConstraints>
	</nationalWithMedianStrip>
	<cityRoad>
		<defaultConstraints>
			<maxSpeed value="13.88889"/>
		</defaultConstraints>
	</cityRoad>
</XMLDefaultRulesDescription>
```

D.5 Capteurs et données

Le modèle XML des capteurs identifie la position des divers capteurs de la simulation, ainsi que les données réelles leur étant associées. Ces dernières peuvent être soit fixes, soit échantillonnées à intervalles réguliers. Dans ce dernier cas, les données figurent dans un fichier ODS tel que celui présenté dans l'annexe C page 36.

Listing D.5 – JAM-FREE : sensors

```
<?xml version="1.0" encoding="UTF-8"?>
<XMLFlowSensorsList>
<!--                            -->
<!-- Definition of the flow sensors at the bounds of the network           -->
<!--                            -->
	<flowSensor id="134">
		<location componentId="R1" end="input" />
		<data>
			<dataFile url="data/134.ods" flowUnit="vehiclesPerMinute" speedUnit="kilometersPerHour" />
		</data>
	</flowSensor>
	<flowSensor id="133">
		<location componentId="R2" end="input" />
		<data>
			<constant initialTrafficFlowInVehiclesPerSecond="0.0" initialAverageSpeedInMetersPerSecond="0.0" />
		</data>
	</flowSensor>
	<flowSensor id="132">
		<location componentId="R5" end="input" />
		<data>
			<dataFile url="data/132.ods" flowUnit="vehiclesPerMinute" speedUnit="kilometersPerHour" />
		</data>
	</flowSensor>
	<flowSensor id="131">
		<location componentId="R8" end="input" />
		<data>
			<constant initialTrafficFlowInVehiclesPerSecond="0.0" initialAverageSpeedInMetersPerSecond="0.0" />
		</data>
	</flowSensor>
	<flowSensor id="130">
		<location componentId="R9" end="input" />
		<data>
			<dataFile url="data/130.ods" flowUnit="vehiclesPerMinute" speedUnit="kilometersPerHour" />
		</data>
	</flowSensor>
	<flowSensor id="129">
		<location componentId="R9" end="output" />
		<data>
			<dataFile url="data/129.ods" flowUnit="vehiclesPerMinute" speedUnit="kilometersPerHour" />
		</data>
	</flowSensor>

	<flowSensor id="Ex1">
		<location componentId="R10" end="output" />
		<data>
			<constant initialTrafficFlowInVehiclesPerSecond="0.0" initialAverageSpeedInMetersPerSecond="0.0" />
		</data>
	</flowSensor>
	<flowSensor id="En1">
		<location componentId="R11" end="input" />
		<data>
			<constant initialTrafficFlowInVehiclesPerSecond="0.0" initialAverageSpeedInMetersPerSecond="0.0" />
		</data>
```

```
      </flowSensor>
</XMLFlowSensorsList>
```

D.6 Décomposition en clusters

Le modèle XML de la décomposition initiale en clusters identifie les clusters de la simulation ainsi que le modèle de flux de trafic routier initialement utilisé en leur sein.

Listing D.6 – JAM-FREE : clusters

```
<?xml version="1.0" encoding="UTF-8"?>
<XMLSimulationInitialStateDescription>
<!--                          -->
<!-- Declaration of CLUSTER 1                 -->
<!--                          -->
       <cluster>
                     <!-- The bounds of the area of the network being modeled by this cluster.     -->
              <bounds>
                     <input sensorId="134" />
                     <input sensorId="En1" />
                     <output sensorId="Ex1" />
                     <output sensorId="129" />
              </bounds>
                     <!-- The alias of the model used to manage the traffic flow inside this cluster.     -->
                     <!-- The alias is declared in the main file of the simulation.          -->
              <trafficFlowModel alias="Micro model with Simple IDM and MOBIL" />
       </cluster>
<!--                          -->
<!-- Declaration of CLUSTER 2 (outside world)              -->
<!--                          -->
       <cluster mutable="false">
                     <!-- The bounds of the area of the network being modeled by this cluster.     -->
              <bounds>
                     <input sensorId="Ex1" />
                     <input sensorId="129" />
                     <output sensorId="134" />
                     <output sensorId="En1" />
              </bounds>
       </cluster>
</XMLSimulationInitialStateDescription>
```

D.7 Modèle microscopique de flux de trafic

Le fichier XML du modèle microscopique de trafic routier fournit l'ensemble des paramètres nécessaires au bon fonctionnement du niveau "Microscopique" (classe de génération de véhicules, paramètres des modèles de suivi et de changement de voie, *etc.*).

Listing D.7 – JAM-FREE : microscopic model

```
<?xml version="1.0" encoding="UTF-8"?>
<XMLParametersMap>
<!--
       Define the class used to generate the 'Cluster controller' agent.
-->
       <param name="clustercontroller.factoryclass"
                     value="fr.lgi2a.jamfree.flowmodel.microscopic.agents.clustercontrollerformicro.AgtMicroClusterControllerFactory" type="string
                           " />

<!-- Define the time range used to measure the effective traffic flow in the microscopic level. -->
       <param name="aggregation.snapshotTimeInterval"     value="2" type="double" />
       <param name="aggregation.snapshotsMaxNumber"     value="10" type="int" />

<!-- Define the values used by the 'Vehicle generator' agents to determine if a vehicle can be safely
       created in the lane. The headwayTime is then used to determine it actual speed, if driving at the
       preferred speed might cause an accident.
-->
       <param name="safeGeneration.headwayTime"      value="1" type="double" />
       <param name="safeGeneration.minimalInterVehicleDistance"  value="2" type="double" />
       <param name="safeGeneration.frontVehicleSpeedAttenuation"  value="0.75" type="double" />

<!-- Define the vehicle factory managing the creation of new vehicles.
 -->
       <param name="vehicleFactoryClass"  value="fr.lgi2a.jamfree.flowmodel.microscopic.libs.vehicles.factories.
              VehicleFactoryWithAccelerationAndLaneChangeModel" type="string" />

<!--
       Define the forward acceleration model used by the vehicles.
```

```
-->
       <param name="forwardAcceleration.className" value="fr.lgi2a.jamfree.flowmodel.microscopic.libs.vehicles.models.forwardacceleration.idm.
             SimpleIDMModel" type="string" />

<!-- Define the 'Vehicle' agents generation parameters used by the above mentioned factory.
 -->
       <param name="vehicleType"      value="car" type="string" />
       <param name="vehicleWeight"     value="1238" type="double" />
       <param name="vehicleHeight"     value="1.53" type="double" />
       <param name="vehicleWidth"      value="1.75" type="double" />
       <param name="vehicleLength"     value="4.13" type="double" />
<!--
       The parameters used to create the Simple IDM Model describing the acceleration behavior of the vehicles.
-->
       <param name="forwardAcceleration.idm_a"   value="3.0" type="double" />
       <param name="forwardAcceleration.idm_b"   value="5.0" type="double" />
       <param name="forwardAcceleration.idm_s0"   value="2.0" type="double" />
       <param name="forwardAcceleration.idm_T"   value="1.5" type="double" />
       <param name="forwardAcceleration.idm_delta"   value="4.0" type="double" />
<!--
       The parameters used to create the lane change model of the vehicle.
-->
<param name="laneChange.delayBeforeCheck"      value="1" type="int" />
<param name="laneChange.safety.className"      value="fr.lgi2a.jamfree.flowmodel.microscopic.libs.vehicles.models.lanechange.safety.
      MobilSafetyModel" type="string" />
<param name="laneChange.safety.safeDeceleration"    value="4.0" type="double" />
<param name="laneChange.safety.safeHeadway"   value="1.0" type="double" />
<params startingWith="laneChange.safety.accelerationAnticipation." sameAs="forwardAcceleration." />
<param name="laneChange.incentive.className"     value="fr.lgi2a.jamfree.flowmodel.microscopic.libs.vehicles.models.lanechange.incentive.
      MobilIncentiveModel" type="string" />
<param name="laneChange.incentive.politeness"     value="0.5" type="double" />
<param name="laneChange.incentive.changeThreshold"    value="0.2" type="double" />
<params startingWith="laneChange.incentive.accelerationAnticipation." sameAs="forwardAcceleration." />
<params startingWith="laneChange.incentive.forwardAcceleration." sameAs="forwardAcceleration." />
<param name="laneChange.speed.className"      value="fr.lgi2a.jamfree.flowmodel.microscopic.libs.vehicles.models.lanechange.speed.StaticSpeedModel"
       type="string" />
<param name="laneChange.speed.lateralSpeed"     value="50.0" type="double" />
</XMLParametersMap>
```

D.8 Modèle à données réelles

Le second modèle de flux de trafic routier utilisé dans cette simulation est un modèle à données réelles. Ce modèle n'effectue pas de simulation à proprement parler. Il se contente de définir le flux à ses capteurs d'entrée et de sortie comme étant égal aux données réelles des capteurs.

Listing D.8 – JAM-FREE : real data model

```
<?xml version="1.0" encoding="UTF-8"?>
<XMLParametersMap>
<!--
       Define the class used to generate the 'Cluster controller' agent.
-->
       <param name="clustercontroller.factoryclass"
                     value="fr.lgi2a.jamfree.flowmodel.realdata.agents.clustercontrollerforrealdata.AgtRealDataClusterControllerFactory" type="
                           string" />
</XMLParametersMap>
```

Annexe E

JAM-FREE Framework

JAM-FREE Framework (JFF) est une interface graphique qui permet aux utilisateurs de gérer simplement des simulations. Des captures d'écran de JFF sont présentées dans les fig. E.1–E.4.

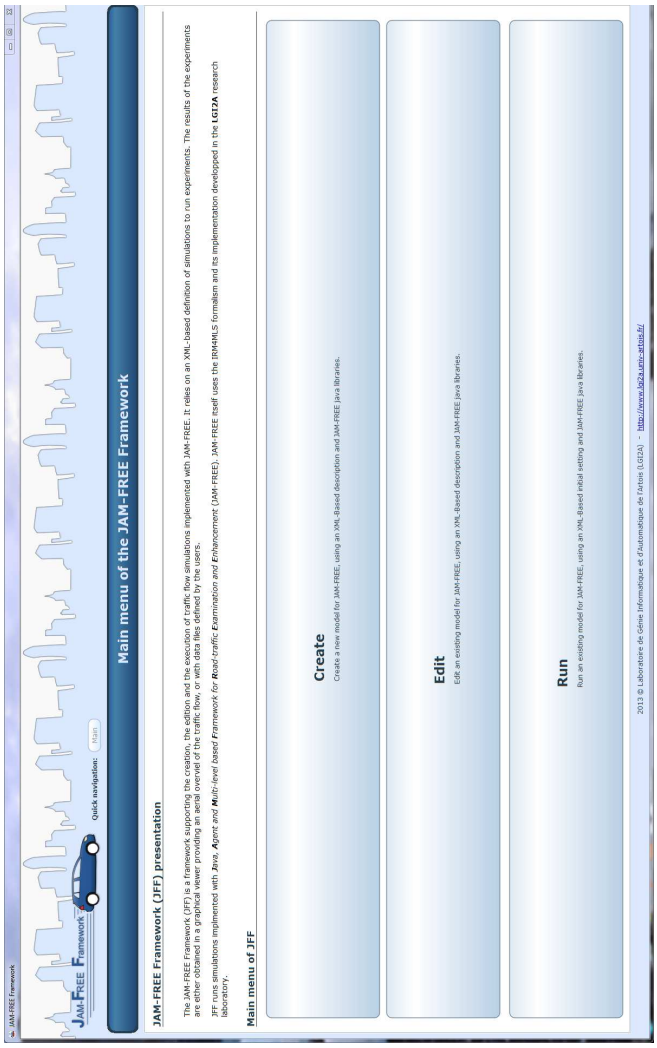


FIGURE E.1 – Screenshot de JFF : menu principal

FIGURE E.2 – Screenshot de JFF : définir les paramètres d'une simulation

FIGURE E.3 – Screenshot de JFF : exécution d'une simulation

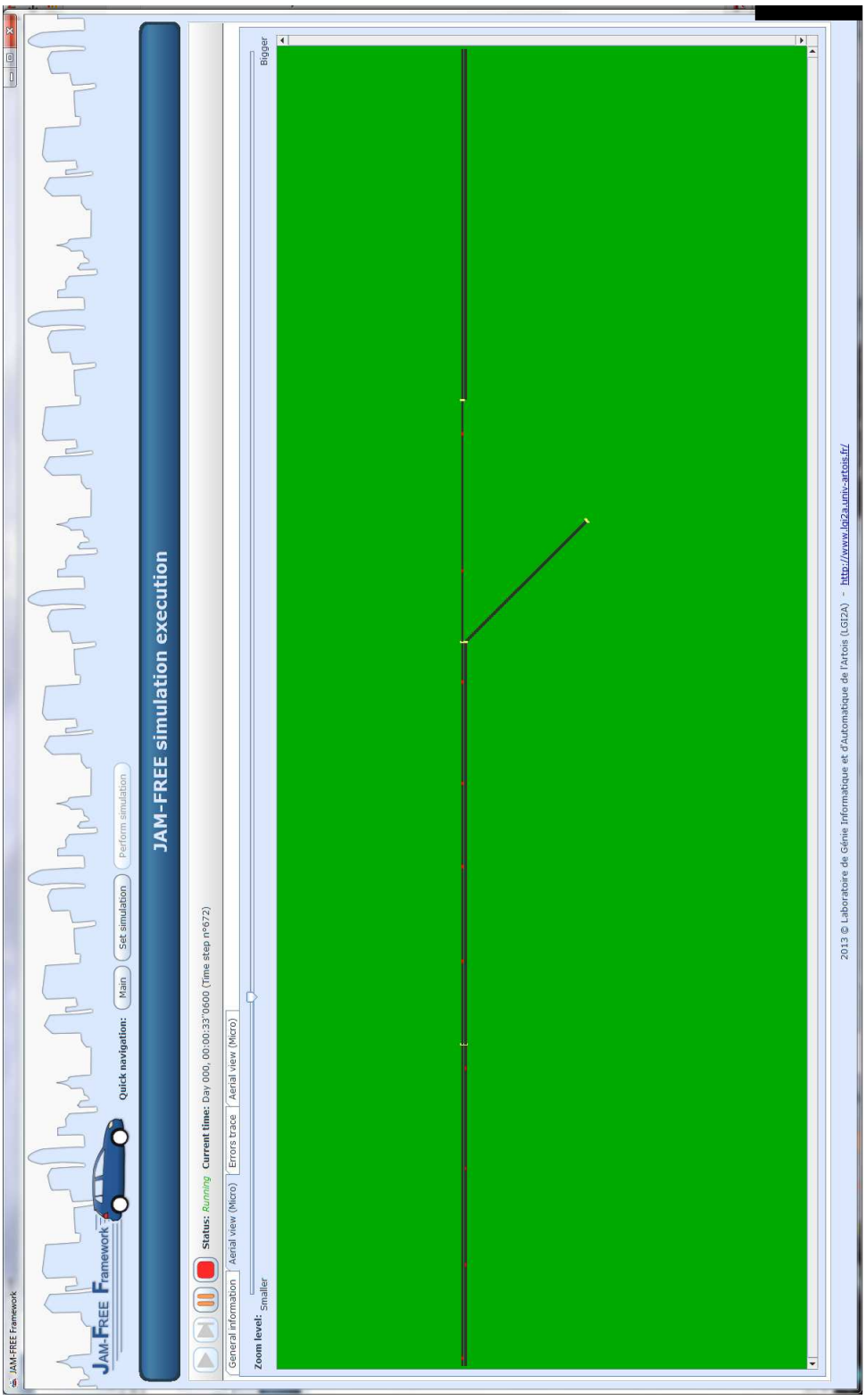


FIGURE E.4 – Screenshot de JFF : exécution d'une simulation(debug mode)

Annexe F

SonarQube

SonarQube est une plate-forme open source pour l'inspection continue (CI) de la qualité du code. Il calcule de nombreux paramètres de qualité telles que le pourcentage de cas couverts par les tests unitaires. Comme les fig. F.1 et F.2 le montrent, SIMILAR obtient d'excellents rapports.

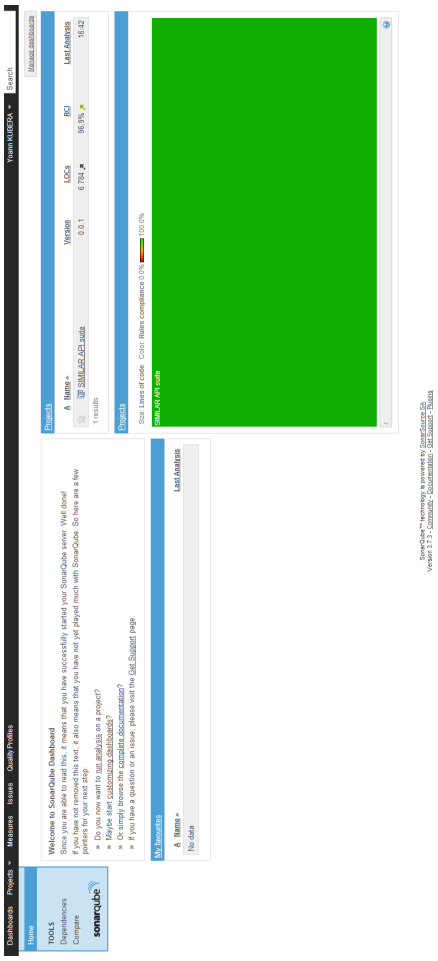


FIGURE F.1 – Screenshot de SonarQube : Ecran principal montrant que SIMILAR est conforme à 96.9% aux meilleures pratiques de développement

FIGURE F.2 – Screenshot de SonarQube : Ecran montrant les différentes métriques

Bibliographie

Bourrel, E. et Lesort, J. (2003). Mixing micro and macro representations of traffic flow : a hybrid model based on the LWR theory. *82th Annual Meeting of the Transportation Research Board*.

El hmam, M. (2006). *Contribution à la modélisation et à la simulation hybride du flux de trafic*. Thèse de doctorat, Université d'Artois.

Espié, S., Gattuso, D. et Galante, F. (2006). Hybrid traffic model coupling macro- and behavioral microsimulation. *In 85th Annual Meeting of Transportation Research Board*.

Ferber, J. et Müller, J.-P. (1996). Influences and reaction : a model of situated multiagent systems. *In 2nd International Conference on Multi-agent systems (ICMAS'96)*, pages 72–79.

Gaud, N., Galland, S., Gechter, F., Hilaire, V. et Koukam, A. (2008). Holonic multilevel simulation of complex systems : Application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory*, 16:1659–1676.

Gil-Quijano, J., Hutzler, G. et Louail, T. (2010). Accroche-toi au niveau, j'enlève l'échelle. éléments d'analyse des aspects multiniveaux dans la simulation à base d'agents. *RIA*, 24(5):625–648.

Gil-Quijano, J., Louail, T. et Hutzler, G. (2012). From biological to urban cells : Lessons from three multilevel agent-based models. *In* Desai, N., Liu, A. et Winikoff, M., éditeurs : *Principles and Practice of Multi-Agent Systems*, volume 7057 de *LNCS*, pages 620–635. Springer.

Kesting, A., Treiber, M. et Helbing, D. (2010). Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 368(1928):4585–4605.

Magne, L., Rabut, S. et Gabard, J. (2000). Towards an hybrid macro-micro traffic flow simulation model. *Proceedings of the INFORMS Salt Lake City String 2000 Conference*.

Mammar, S.and Lebacque, J. et Haj-Salem, H. (2006). Hybrid model based on second-order traffic model. *In Proc. of TRB 85th Annual Meeting*, numéro 06-2160.

Michel, F. (2007a). The IRM4S model : the influence/reaction principle for multiagent based simulation. *In Proc. of 6th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 1–3.

Michel, F. (2007b). Le modèle IRM4S. de l'utilisation des notions d'influence et de réaction pour la simulation de systèmes multi-agents. *RIA*, 21:757–779.

Morvan, G. (2013). Multi-level agent-based modeling - a literature survey. *CoRR*, abs/1205.0561.

Morvan, G. et Jolly, D. (2012). Multi-level agent-based modeling with the Influence Reaction principle. *CoRR*, abs/1204.0634.

Morvan, G. et Kubera, Y. (2014a). On time and consistency in multi-agent-based simulations. Working paper.

Morvan, G. et Kubera, Y. (2014b). SIMILAR : Simulations with multi-level agents and reactions. Working paper.

Morvan, G., Veremme, A. et Dupont, D. (2011). IRM4MLS : the influence reaction model for multi-level simulation. *In Multi-Agent-Based Simulation XI*, volume 6532 de *LNCS*, pages 16–27. Springer.

Picault, S. et Mathieu, P. (2011). An interaction-oriented model for multi-scale simulation. *In* Walsh, T., éditeur : *Twenty-Second International Joint Conference on Artificial Intelligence*, pages 332–337, Barcelona, Spain. AAAI Press.

Poschinger, A., Kates, R. et Keller, H. (2002). Coupling of concurrent macroscopic and microscopic traffic flow models using hybrid stochastic and deterministic disaggregation. *Transportation and Traffic Theory for the 21st century.*

Sétra (2010). Identification et localisation sur le reseau routier national. Rapport technique, Sétra, 110 rue de Paris 77171 Sourdun - France.

Sewall, J., Wilkie, D. et Lin, M. C. (2011). Interactive hybrid simulation of large-scale traffic. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)*, 30(6).

Soyez, J.-B., Morvan, G., Dupont, D. et Merzouki, R. (2013). A methodology to engineer and validate dynamic multi-level multi-agent based simulations. *In Multi-Agent-Based Simulation XIII*, volume 7838 de *LNCS*, pages 130–142. Springer.

Vo, A. (2012). *An operational architecture to handle multiple levels of representation in agent-based models.* Thèse de doctorat, Université Paris VI.

Vo, D.-A., Drogoul, A. et Zucker, J.-D. (2012a). An operational meta-model for handling multiple scales in agent-based simulations. *In International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 1–6, Ho Chi Minh City. IEEE.

Vo, D.-A., Drogoul, A., Zucker, J.-D. et Ho, T.-V. (2012b). A modelling language to represent and specify emerging structures in agent-based model. *In* Desai, N., Liu, A. et Winikoff, M., éditeurs : *Principles and Practice of Multi-Agent Systems*, volume 7057 de *LNCS*, pages 212–227. Springer.